Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2014

Luís Carlos
Barata Duarte Guedes

Transmissão Oportunística de Informação em Redes Veiculares

Opportunistic Transmission of Information in Vehicular Networks

Universidade de Aveiro
Departamento de
Electrónica, Telecomunicações e Informática
2014

Luís Carlos
Barata Duarte Guedes

Transmissão Oportunística de Informação em Redes Veiculares

Opportunistic Transmission of Information in Vehicular Networks

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica da Professora Doutora Susana Isabel Barreto de Miranda Sargento, Professora Associada do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e coorientação de Tiago Condeixa, Engenheiro na VeniamWorks.

o júri / the jury

presidente / president	**Professor Doutor Rui Luís Andrade Aguiar** Professor Associado com Agregação no Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro
Arguente	**Professor Doutor Joel José Puga Coelho Rodrigues** Professor Auxiliar no Departamento de Informática da Faculdade de Engenharia da Universidade da Beira Interior
Orientadora	**Professora Doutora Susana Isabel Barreto de Miranda Sargento** Professora Associada no Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Agradeço à minha família, em especial aos meus pais e irmã pelo grande apoio e motivação ao longo de todo o meu percurso académico. Também à Ana Guimarães pela presença, dedicação e incentivo nos momentos mais difíceis.

Um agradecimento especial à Professora Susana Sargento por me ter proporcionado as condições necessárias para a execução deste projecto e por permitir a minha integração num centro de investigação de tão elevada qualidade e exigência. Também pela oportunidade que me deu no estabelecimento do meu primeiro contacto com o ambiente empresarial. Ainda pela sua orientação, coordenação e dedicação demonstradas ao longo de todo o projecto.

Agradeço a todos os colegas e amigos que me acompanharam ao longo do curso bem como àqueles que conheci no Instituto de Telecomunicações, em especial ao Romeu Monteiro, ao Lucas Guardalben e ao Bruno Tavares pela disponibilidade, companheirismo e ajuda no desenvolvimento deste trabalho. Todos eles contribuiram para que esta árdua caminhada fosse concluída com sucesso.

Agradeço ao Tiago Condeixa, ao Filipe Neves e ao Diogo Lopes pela disponibilidade que sempre manifestaram e pela ajuda na integração e realização dos testes experimentais na testbed do Porto de Leixões.

Por último, mas não menos importante, um agradecimento particular ao João Mendes e ao João Azevedo por toda a companhia e partilha nos bons e menos bons momentos, ao longo de cinco anos, em Aveiro e em Erasmus. Também ao Fábio Martins, João Aparício, Rafael Gomes, Duarte Santos, André Ferraz e Gonçalo Pardal pelo companheirismo e momentos de boa disposição, no laboratório.

Resumo

O desenvolvimento na área das telecomunicações e, mais particularmente, nas comunicações sem-fios tem sido um dos traços mais marcantes do mundo actual. A globalização só tem sido possível graças à evolução dos meios de comunicação que cada vez mais permitem satisfazer a constante necessidade das pessoas estarem sempre ligadas, qualquer que seja o ambiente em que se encontrem.

As redes veiculares têm sido uma das áreas de elevado interesse na evolução das tecnologias. Esse interesse tem-se manifestado tanto ao nível da investigação como ao nível do desenvolvimento da indústria automóvel que tem produzido veículos cada vez mais equipados com novas tecnologias. Prevê-se que a comunicação em redes veiculares permitam não só a comunicação entre os veículos, mas também uma condução mais confortável e segura, tornando a experiência dos utilizadores deste tipo de redes mais rica e estimulante. As características específicas das redes veiculares, nomeadamente a elevada mobilidade, rotas imprevisíveis, topologia dinâmica e a consequente e constante perda de conectividade, tornam-se um desafio que tem motivado estudos no sentido de se encontrarem soluções para essas limitações.

O trabalho desenvolvido para esta dissertação insere-se na área das Vehicular Ad-hoc NETworks (VANETs) e baseia-se nas Delay and Disruption Tolerant Networks (DTNs). Com este projecto, identificado como "Transmissão Oportunística de Informação em Redes Veiculares", pretende-se estudar a comunicação e envio de informação nas redes que permitem uma comunicação com atrasos e disrupções. Para o efeito é estudado o desempenho de mecanismos de DTN nestas redes.

Neste trabalho é utilizada a implementação IBR-DTN para testar DTN nas redes veiculares. Esta implementação mostrou, em trabalhos anteriores, ser aquela que apresenta melhor desempenho face a outras que existem. O estudo envolveu, numa fase inicial, a leitura e análise de código da implementação para que fosse possível adicionar instruções que permitissem observar o comportamento da implementação nos diversos testes realizados, bem como a correcção de erros da implementação.

Na primeira fase, em laboratório, com nós fixos e num ambiente controlado, foram realizados vários cenários que mostram as situações possíveis que um nó pode encontrar: transferência direta com e sem atraso, transferência indirecta (multi-hop) e transferência indirecta com atraso que corresponde ao armazenamento e transporte dos bundles (conjunto de informação) até ao próximo nó. A partir da análise da informação recolhida e observação dos gráficos obtidos foi possível verificar o correcto funcionamento da implementação nos equipamentos de comunicação entre veículos. Ainda em laboratório foi construída uma rede heterogénea com diversos dispositivos (servidores, NetRiders, Single Board Computers (SBCs), tablet, Raspberry Pi e Macbook) com o objectivo de mostrar a integração da implementação IBR-DTN e as suas extensões em diferentes equipamentos.

Neste teste foram enviados ficheiros entre estes dispositivos, os quais foram recebidos correctamente nos nós definidos como destino.
Depois de testar e certificar que tudo funcionava em laboratório, a mesma implementação foi transferida para uma testbed com 25 veículos e 3 infraestruturas fixas, no porto de Leixões. Nesta testbed foram testados diversos protocolos de encaminhamento DTN de forma a verificar qual apresentava melhor desempenho na taxa de entrega dos bundles e da informação recolhida (os ficheiros de log foram também entregues através de DTN) das On-Board Units (OBUs) para o servidor, localizado na Internet. O protocolo com rotas estáticas para as Road Side Units (RSUs) demonstrou uma melhor eficiência em relação aos restantes devido ao facto de esta rede estar bem coberta e de não existir uma relação entre o histórico de contactos e a probabilidade de os veículos se encontrarem novamente.

Abstract

The development in telecommunications and particularly in wireless communications has been one of the most striking features of the contemporary world. The globalization only has been possible thanks to the evolution of communication technologies which increasingly have allowed to satisfy the constant people's needs of being "always connected" whatever the environment where they are.

Concerning the evolution of technologies, vehicular networks have been one of the areas of great interest. This interest has been manifested both in research and in the development of the automotive industry that has produced innovative vehicles which are more and more equipped with new technologies. It is expected that communication in vehicular networks enable not only the communication between vehicles, but also a more comfortable and safe driving, making the user's experience of this type of networks richer and stimulating. The specific characteristics of vehicular networks, namely the high mobility, unpredictable routes, dynamic topology and the consequent and constant loss of connectivity, have been a challenge that has motivated studies to find solutions to these limitations.

The work carried out for this dissertation is in the area of Vehicular Ad-hoc Networks (VANETs) and it is based on the Delay and Disruption Tolerant Networks (DTNs). With this project, identified as "Opportunistic Transmission of Information in Vehicular Networks", we aim to study the communication and transmission of information in these networks which do not allow communication without delays and disruptions. For this purpose it is studied the performance of DTN mechanisms in these networks.

In this work it is used the implementation IBR-DTN to test DTN in VANETs. This implementation showed, in previous works, to be the one that presents the best performance comparing it with other existing implementations. The study involved, in an initial phase, reading and analyzing the implementation code so that it was possible to add instructions that allowed to observe the behavior of the implementation in the several tests carried out, as well as the correction of the bugs in the implementation.

In the first phase, in laboratory, with fixed nodes and in a controlled environment, several scenarios were created to simulate the possible situations a node can meet: direct transfer with and without delay, indirect transfer (multi-hop) and indirect transfer with delay which corresponds to the store and transport of the bundles (set of information) until the next node. From the analysis of the collected information and observing the corresponding graphs, it was possible to observe that the implementation was working properly in the vehicles equipment for communication. Still in laboratory it was built an heterogeneous network with several devices (servers, NetRiders, Single Board Computers (SBCs), tablet, Raspberry Pi e Macbook) to show the integration of the IBR-DTN implementation and its extension in different equipments.

During this test several files were sent among these devices, which were correctly received in the nodes previously defined as destination nodes. After testing and checking that everything was working properly in the laboratory, the same implementation was transferred to a testbed with 25 vehicles and 3 fixed infrastructures in Leixões harbor. In this testbed several DTN routing protocols were tested in order to check which of them showed better performance in the delivery rate of the bundles and of the collected information (the log files were also delivered via DTN) from the On-Board Units (OBUs) to the server, located in the Internet. The routing protocol with static routes to the Road Side Units (RSUs) proved a better efficiency compared to the other protocols. This was due to the the fact that this network is well covered with RSUs, and there is no relation between the historic of contacts and the probability that the vehicles will meet again in the future.

Contents

List of Figures

List of Tables

Acronyms

ACK	Acknowledgement
ADU	Application Data Unit
AP	Access Points
AU	Application Unit
BAB	Bundle Authentication Block
BER	bit-error rate
BP	Bundle Protocol
BSP	Bundle Security Protocol
BSS	Bundle Streaming Service
CCH	Control Channel
CCSDS	Consultative Committee for Space Data Systems
CGR	Contract Graph Routing
CLA	Convergence Layer Adapters
CoS	Classes of Service
CPU	Central Processing Unit
C2CCC	Car-to-Car Communication Consortium
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNS	Domain Name System
DSRC	Dedicated short-range communications
DTN	Delay Tolerant Network
DTNRG	Delay-Tolerant Networking Research Group
EID	Endpoint Identifiers

E2E	End-to-End
EMMA	Environmental Monitoring in Metropolitan Areas
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
GUI	Graphical User Interface
HD	High-definition
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPN	InterPlanetary Internet
IPND	Internet Protocol Neighbor Discovery
IPv6	Internet Protocol version 6
ISA	Instruction Set Architecture
IT	Instituto de Telecomunicaçoes
ITS	Intelligent Transportation Systems
LAN	Local Area Networks
LTE	Long Term Evolution
LTP	Licklider Transmission Protocol
MAC	Medium Access Control
MANET	Mobile Ad-hoc Networks
M2M	Machine-to-Machine
NTP	Network Time Protocol
OBU	On-Board Units
OS	Operative System
PAN	Personal Area Network
PDA	Personal Digital Assistant
PDU	Protocol Data Unit

PROPHET	Probabilist ROuting Protocol using History of Encounters and Transitivity
PTPD	Precision Time Protocol daemon
P2P	Peer-to-Peer
RAM	Random Access Memory
RAPID	Resource Allocation Protocol for Intentional DTN
RPi	Raspberry Pi
RSU	Road Side Units
SBC	Single Board Computer
SCH	Service Channel
SDNV	Self-Delimiting Numeric Value
SMS	Short Message Service
TCP	Transmission Control Protocol
TTL	Time-to-Live
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
URI	Universal Resource Identifiers
USB	Universal Serial Bus
UTC	Coordinated Universal Time
VANET	Vehicular Ad-hoc Network
VDTN	Vehicular Delay Tolerant Network
VM	Virtual Machine
VN	Vehicular Networks
VSAT	Very-small-aperture terminal
V2I	Vehicle-to-infrastructure
V2R	Vehicle-to-Road
V2V	Vehicle-to-vehicle
WAVE	Wireless Access in Vehicular Environments
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
ZOR	Zone-of-relevance

Chapter 1

Introduction

The main aim of this dissertation is to present the work that was developed in real environments to introduce the concept of "DTN" in vehicular networks.

With the work performed it is expected to evaluate the behavior of DTNs and to know if they may be used in vehicular networks, ensuring that all the tests are performed in a real vehicular platform.

In this first chapter it is presented the motivation for this work, its main objectives and a brief description of the document organization.

1.1 Context and Motivation

Communications have played a crucial role in society. Today the number of things connected to the Internet is larger than the number of people in the world [19]. In the last years we have been observing a huge development in wireless networks which are being more and more used in communications. They will be also very used in the future (according to Cisco's data [1], the global traffic of mobile data will increase 10 times in the next four years) as it is shown is Figure 1.1. Wireless networks represent a more and more increasing part of telecommunications networks which are characterized by speed, cost, reliability and mobility.

The automotive industry has been changing the characteristics of vehicles by incorporating a lot of new services, such as navigating systems (GPS), pre-crash sensing, velocity control systems, on board computers and other technologies that provide more safety to the drivers, as well as much more comfort for a better driving experience. In the future, surely the vehicles will also incorporate radio systems in order to communicate with each other and so creating a Vehicular Ad-hoc NETwork (VANET).

The mobile nodes in a VANET (vehicles) are highly mobile, and their mobility is constrained by the underlying road network topology. VANETs are characterized by a very challenging and dynamic environment, since they combine a fixed infrastructure (RSUs) with Ad-hoc communications among vehicles. The communication between nodes that are not in the range of transmission of the radio is made, whenever possible, by multi-hop through the intermediate node contribution.

As this type of networks involve high mobility, mobile nodes may often lose the connection. With DTNs they can be interrupted at any moment thanks to *store*, *carry* and *forward* mechanism, which allow the disruption of the connections between nodes.

This kind of networks require a good management of the connections between the nodes whenever there is a connection with the purpose of maximizing the global connectivity. The communication can be established between vehicles and infrastructures along the road and with IEEE 802.11 a/g/n/p (Wi-Fi) hotspots. There has already been some research work in DTNs in order to develop this kind of networks which has been essential to enlarge and consolidate the concept of DTN in VANETs.

DTN has become very important to face the main challenges of VANETs. On the one hand, because of its instability and connection losses which do not allow the right communication among vehicles. On the other hand, DTN may also be used to transfer non-urgent information which is created by sensors from *smart cities* platform.

After the work developed in a previous thesis in our group, in which some existing DTN implementations were studied, there was a strong necessity to develop the concept and applications of DTN as well as to implement and test it on VANETs. The main goal of this project is to enable the data processing in a real-world vehicular mesh network capable of retrieving data collected by large number of sensor nodes integrated in a testbed. Another aim is to show the integration of different devices with DTN software in an heterogeneous network. Special attention will be given to energy-aware machine-to-machine protocols and to the store-and-forward mechanisms.

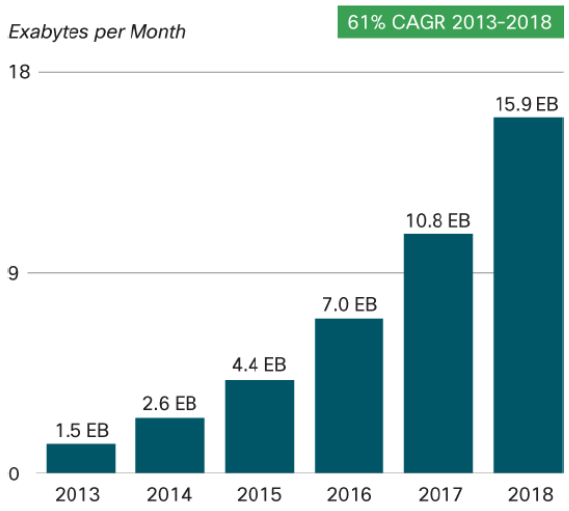


Figure 1.1: Cisco Forecasts - Mobile Data Traffic by 2018 [1].

1.2 Objectives

This dissertation has as its main aim to explore the implementation of IBR-DTN and the transmission of information that is tolerant to delays in vehicular networks, and the several DTN dissemination protocols since not all services in a telecommunications network are urgent and do not need to be transmitted immediately.

Some examples of delay tolerant applications include: sharing music between vehicles, tourist reception of information about a city or specific location, and gathering information from vehicle conditions. At the network layer there are mechanisms that

allow network elements (vehicles) to store the information while there are no other vehicles connected, and then, they send the information opportunistically when a vehicle is under the range of connection.

In this dissertation, the mechanisms above mentioned will be tested in the laboratory and in the road (in a platform of tests), to evaluate how much they can increase the transmission of data, as well as the transmission of information from sensors, considering both Vehicle-to-vehicle (V2V) and Vehicle-to-infrastructure (V2I) communications. Moreover, in a vehicular network it is important to have a solution that minimizes the resource usage of the equipment in the vehicle to store information, to propagate urgent information in useful time and avoid the spreading of irrelevant information.

In a more concrete approach, the objectives of this Dissertation are the following:

- to change the IBR-DTN source code to extract log data necessary for proper DTN evaluation.
- to install, check and debug the implementation in the vehicular equipment.
- to evaluate the performance of IBR-DTN in laboratory tests.
- to show the integration of IBR-DTN in an heterogeneous network.
- to develop software to analyse the data of the several tests.
- to extend the tests to a real platform in order to test several routing mechanisms.
- to update the implementation to a test platform (Leixões Harbor, Oporto).
- to evaluate and to improve the performance of IBR-DTN with tests in the Harbor.
- to evaluate the performance of different routing protocols tested.
- to evaluate the general performance comparing it to the existing architecture.

1.3 Contributions

This Dissertation provided the following contributions:

- It was changed the IBR-DTN package with delay tolerant protocols to work with IEEE 802.11p and vehicular networks;
- Real platforms were developed, both in the lab and in the harbor of Leixões with real vehicles and road side units;
- Real experiments were performed in both platforms, and performance metrics were gathered and analysed for several delay-tolerant dissemination protocols;
- The performed experiments provide insights of how delay-tolerant dissemination approaches perform in vehicular environments, which guide on the required characteristics of such approach.
- A paper entitled "Lessons From a Real Wireless Network Deployment of Delay-Tolerant Networking" is submitted to IEEE ICC Workshop on Dependable Vehicular Communications (DVC) 2015.

1.4 Dissertation Structure

The present dissertation is structured in six fundamental chapters as follows:

- Chapter 1 - Introduction: provides the contextualization of this Dissertation presenting the motivation, the proposed objectives and the document structure.
- Chapter 2 - State of Art: presents the state of the art, an overview of the main concepts in vehicular networks, the mobility protocols and their possible application on VANETs.
- Chapter 3 - DTN Implementation: gives the reader an overview of DTN implementations and, in specific, the used implementation IBR-DTN which is briefly described: its architectures, features and applications.
- Chapter 4 - Integration in experimental testbed: provides some information about the equipment used in the experimental tests. It presents also the developed work, difficulties and problems found in IBR-DTN for all tested scenarios.
- Chapter 5 - Results: presents the tests and the results obtained for the different scenarios tested both in laboratory and in Leixões harbor testbed.
- Chapter 6 - Conclusion and Future Work: summarize and conclude all work performed along this dissertation, as well as, suggest some improvements as future work, in order to develop DTNs in this type of networks.

Chapter 2

State of Art

2.1 Introduction

In order to fulfill the aims of the work presented for this dissertation, this chapter will introduce some concepts related to the developed work. It also provides an overview of vehicular networks and the related work in terms of VANET routing, DTN and vehicular DTNs.

The chapter is divided into three main sections:

Section 2.2 – presents the definition and basic concepts of VANETs as well as their architecture, challenges and routing.

Section 2.3 – introduces the concept of DTN focusing on its characteristics, architecture, routing and main applications.

Section 2.3 – presents some DTN applications used in Vehicular Delay Tolerant Networks (VDTNs), and focusing on several projects.

2.2 Vehicular Networks

Nowadays, permanent and unlimited connectivity to the Internet is available for a large number of mobile and fixed devices. Vehicles traveling within cities and along highways are regarded as the most probable candidates for integration into next generation mobile networks. Vehicular networks are a new class of wireless networks and have been the focus of an increasing interest in the research community in the last few years. Not only the developments in wireless communications, but also the evolution and massification of Wireless Local Area Networks (WLANs), allow the increase of the technology in the automotive industry, providing the passengers more safety and comfort. As we can see in Figure 2.1, vehicles can be from any nature: private, public service (police cars and ambulances), public transports (buses and taxis).

Vehicular Ad-Hoc Networks, also known as VANETs, are a particular class of the Mobile Ad-hoc Networks (MANET). VANETs are characterized by a set of vehicles that communicate with each other, or with external and nearby fixed equipments placed on roads. Each node has the responsibility of forwarding data to other nodes. In order to send, transmit or receive information on the network, vehicles need to be equipped with On-Board Units (OBU). On the other hand, the infrastructures along the road, Road Side Units (RSU), are responsible for connecting foreign networks with

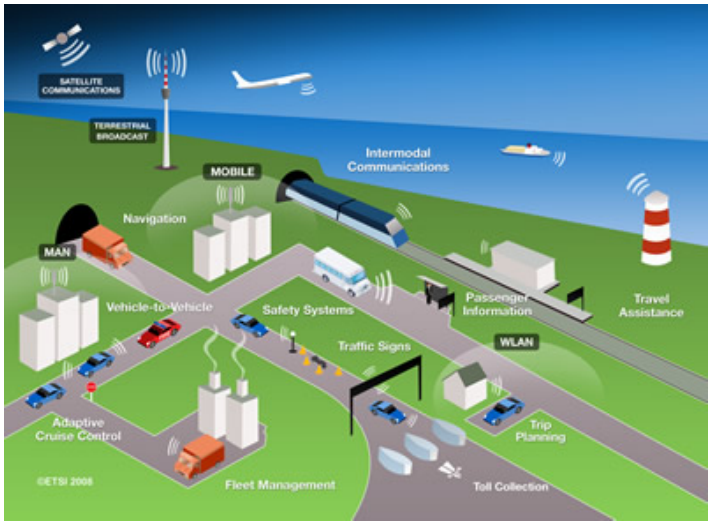


Figure 2.1: The Future of Intelligent Transport Systems [2].

the OBUs and establishing connections to the Internet. Thus, the communication can be established between vehicles (V2V communication) and between vehicles and infrastructure (V2I communication), as it is shown in Figure 2.2.

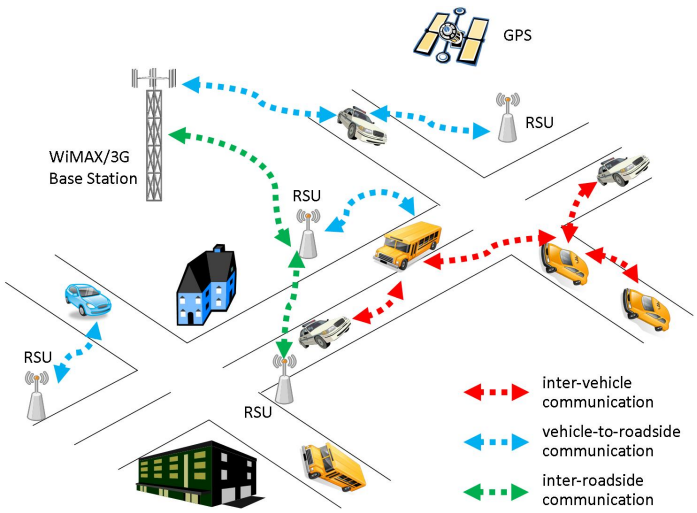


Figure 2.2: Vehicular Network communication types [3].

2.2.1 Main Characteristics and Challenges of Vehicular Networks

Vehicular networks have characteristics and inherent behaviors that distinguish them from other types of mobile networks. Compared with other types of mobile networks, vehicular networks come with attractive features [20]:

- "Unlimited" batteries: The battery of mobile devices is usually not a restriction in vehicular networks, since the node (vehicle) may provide continuous energy for computation and communication between devices.

- Larger computational capacity: Operating vehicles can support significant communication, computing and sensing capabilities.

- Predicted Mobility: Vehicles tend to have predictable movements which are, generally, limited to the roads. The information about the roads may be provided by the positioning systems based technologies like Global Positioning System (GPS), giving the instantaneous and average speed as well as its trajectory.

Vehicular networks have also great challenges:

- Potentially large scale: VANETs are extensible over the entire road network.

- High mobility and dynamic topology: The environment where VANETs operate is extremely dynamic. Due to the high speed of vehicles movement, the topology of VANETs is always changing.

- Frequently disconnected network: Loss of connectivity in the VANETs is frequent specially when the vehicle density is low.

- Various communication environments: VANETs are usually operated in two typical communication environments: highway traffic scenarios with a low number of obstacles in the radio communications, and city scenarios with buildings, trees and other obstacles.

2.2.2 Basic Concepts

This section presents base concepts of VANETs.

2.2.2.1 On-Board Equipment

Vehicles are a crucial part of VANETs and must be eqquiped with OBUs. Devices of this type are composed by the following items:

- a Central Processing Unit (CPU) which implements the application and the communication protocols.

- a wireless transceiver to transmit and receive data to and from vehicles in the neighborhood or infrastructure.

- a GPS receiver to provide accurate positioning and time synchronization data.

- sensors to measure several parameters to be transmitted to other nodes.

- input/output interface which allows human interaction with the system.

2.2.2.2 Addressing

According to [21], most vehicular networks may be classified as Ad-hoc networks. This way, the addressing schemes of Ad-hoc networks may be used in vehicular networks. There can be two types of addressing:

- **Fixed addressing:** each node has a fixed address at the moment it joins the network and uses it while part of the network. This is the most common mechanism for addressing in the Internet. Most Ad-hoc networking applications and protocols assume a fixed addressing scheme.

- **Geographical addressing:** each node is characterized by its geographical position. So, every time the node moves, its address changes. Additional attributes may be used by a group of target vehicles like the direction of movement, road identifier (number or name), type of vehicle, some physical characteristics or even characteristics of the driver.

2.2.2.3 Data Dissemination

One of the main aim of applications in vehicular networks is to disseminate data between vehicles. As shown in the figure 2.3, data dissemination can be single-hop or multi-hop. The first, (fig.2.3(a)), is usually implemented with broadcast on the Medium Access Control (MAC) layer, where the vehicle A can send messages only to the cars which are in its transmission range. In the second scenario, (fig.2.3(b)), data should be transmitted in several hops, where intermediate vehicles work as relays as it is the case of vehicle C, which relays the message emitted from vehicle A. Thus, the multi-hop system requires a network layer capable of multi-hop routing. There are also hybrid variants in which data is disseminated in multi-hop to the closest base station which transmits the data to vehicles in single-hop.

Data can be disseminated in unicast, multicast or broadcast. In unicast, there is just one sender and one receiver. As the scheme of addressing is not relevant, it can be used fixed or geographical addresses.

In multicast, there is just one sender but several receivers. In vehicular networks, many applications related to public safety require data to be disseminated in a specific area to all vehicles that are driving in a specific direction: this corresponds to a multicast group.

Broadcast disseminates data to *all* vehicles. It can spread the messages to the whole world, but it is usually only performed in a specific area, called Zone-of-relevance (ZOR).

2.2.2.4 Network Access Technologies

Nowadays, all communication standards that may be used as access networks for VANETs have advantages and disadvantages depending on the type of application and the scenario. In vehicular communications, fast and slow fading can be expected to bring problems to the radio channel both in the transmitter and in the receiver.

IEEE 802.11 is the most used wireless technology in Wireless Local Area Network (WLAN). So it is the most common technology in vehicular networks. The bit-error

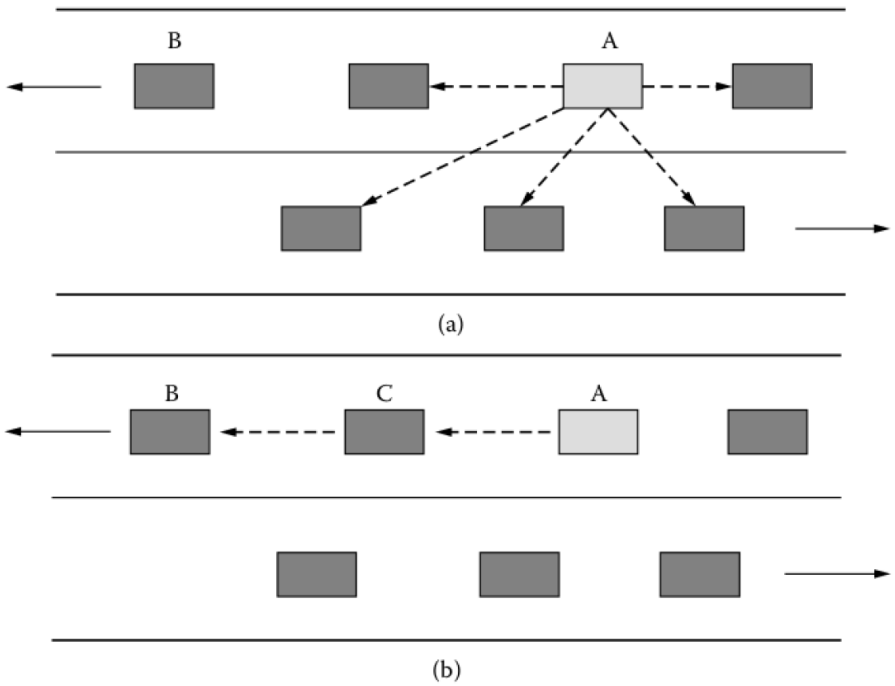


Figure 2.3: Single-hop(a) and multihop(b) data dissemination

rate (BER) for this standard can be very high [22], imposing significant difficulties in higher layers. The problem can be mapped to the design of 802.11a that was developed for Local Area Networks (LAN) with no or low mobility. In vehicular networks, mobility is very high, and the techniques designed for low mobility cannot handle properly the resulting fast fading.

Thus, a new standard - IEEE 802.11p - was developed for vehicular communication. It has better responses against fading, increasing the range with lower losses. The main properties of this standard are described in the next subsection.

2.2.3 IEEE 802.11p

For a future implementation of an efficient VANET architecture, two Institute of Electrical and Electronics Engineers (IEEE) groups (p and 1609) formed in 2004 to add support for Intelligent Transportation Systems (ITS) have developed a standard for the wireless access in vehicular environments known as Wireless Access in Vehicular Environments (WAVE).

The WAVE protocol stack consists in IEEE 802.11p and IEEE1609 [23][24] standards. The IEEE 802.11p standard handles MAC and physical layer functionalities, while IEEE1609 deals with upper layer properties [25].

The standard IEEE 802.11p technology defines mechanisms to be used in high speed radio environments, where the physical layer properties are rapidly changing and where very short duration communications are required as it happens with cars and trucks. It is able to perform communication between vehicles, both with emergency and entertainment services. In these environments, the 802.11p improvements to the previous standards enable robust and reliable V2V and V2I communications, rapidly changing

multipath conditions, and allow for establishing a link and exchange data in very short times (less than 100 ms) [26][27]. This standard provides the minimum set of specifications required to ensure interoperability between wireless devices while communicating in potentially quickly changing communication environments, and in situations where transactions must be completed in time frames as small as the minimum possible with infrastructure or vehicles.

IEEE 802.11p [26] operates in the 5.835-5.925 GHz range and is divided into seven channels of 10 MHz each.

An example of the usage of these networks in the applications referred above is in the figure 2.4 that shows the United States of America Dedicated short-range communications (DSRC) spectrum. It is structured into seven 10MHz wide channels. The channel 178 is the control channel, restricted to safety communications. The two channels at the ends of the spectrum band are reserved for special use, while the rest of the spectrum is available for both safety and non-safety usage.

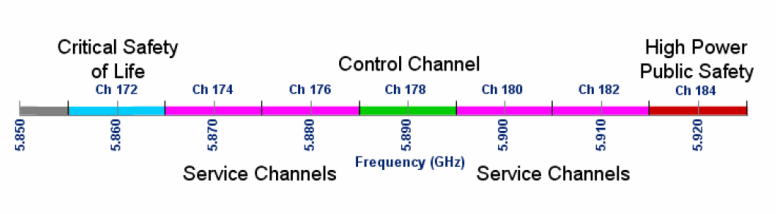


Figure 2.4: US DSRC spectrum [4]

2.2.4 Architecture

In the last years, enhancements in wireless technologies have allowed to implement a vehicular architecture in rural, city and road environments. This architecture may be established according to figure 2.5 and is divided into three types:

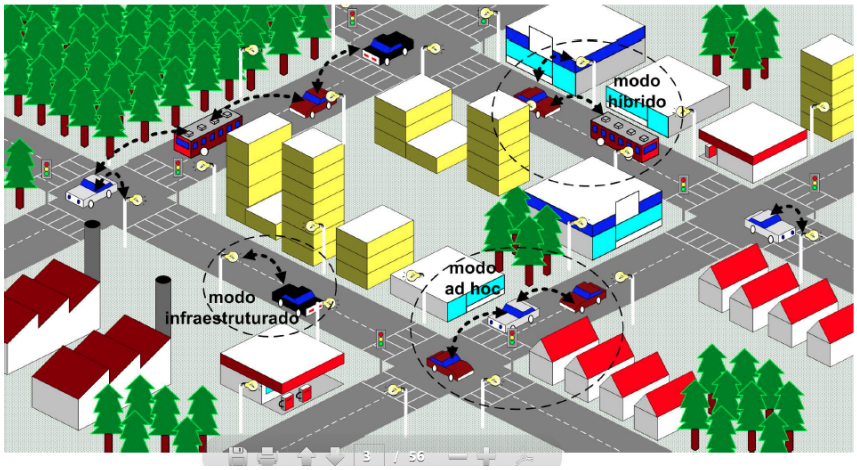


Figure 2.5: VANETs Architecture [5].

- **Pure *Ad-hoc* network - Vehicle-to-Vehicle:** the vehicles communicate with each other without any support of external elements. Each vehicle behaves as a *router*, being responsible for forwarding messages. This is the simplest scenario in vehicular networks because there is no infrastructure, but it is, at the same time, a drawback because the connectivity in the networks rely on the density and pattern of vehicle's mobility.

- **Infrastructured - Vehicle-to-Infrastructure:** it was created to overcome the problems of the previous approach. In this architecture there are static nodes distributed along roads. These nodes act like Access Points (AP) and centralize all the traffic from the network. The use of infrastructure increases connectivity and allows communication with other networks, for example, the Internet.

- **Hybrid - Vehicle-to-Road:** it corresponds to an intermediate solution between Ad-hoc and infrastructured types. This architecture minimizes the number of RSUs used to increase the connectivity of the network and to provide services. In this type of networks, the vehicles may communicate with the infrastructure either through single or multi-hop, according to the positions of the vehicles in relation to the infrastructure.

The Car-to-Car Communication Consortium (C2CCC) proposed a reference architecture for VANET[s], which can be divided in three different domains: in-vehicle, Ad-hoc and infrastructured, as figure 2.6 shows:

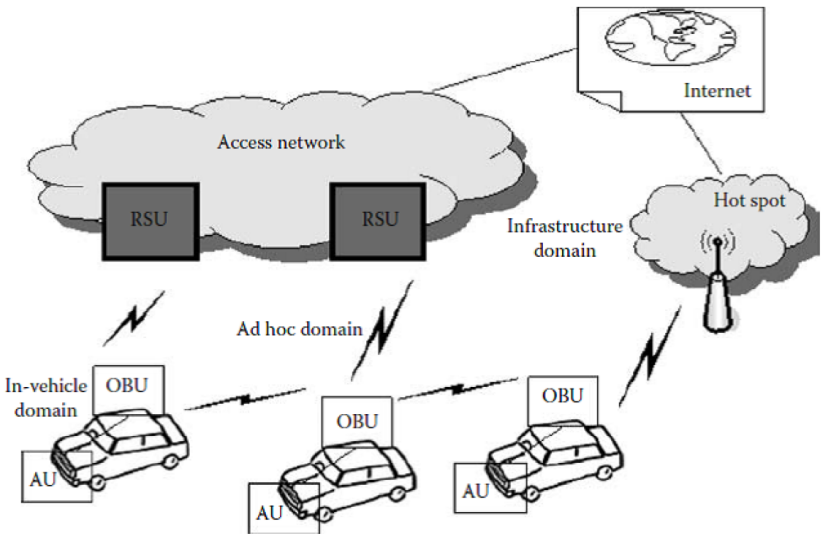


Figure 2.6: VANETs C2CCC Reference Architecture [6]

- **In-vehicle domain:** it refers to a local network inside each vehicle that is composed by two units: OBU, which is described ahead, and one or more Application Unit (AU)s. The OBU is the device which goes into the vehicle and has the capability to communicate (wired or wireless), while the AU is a device that executes

applications, using the communication capabilities of the OBU. An AU may be integrated in the vehicle being permanently connected to the OBU or, on the other hand, it can be portable as a Smartphone which can be intermittently connected to the OBU.

- **Ad-hoc domain:** it is a network composed by vehicles equipped with mobile and fixed nodes, OBUs and RSUs respectively. The OBUs from different vehicles form a VANET and may be connected to an infrastructured network making possible the access to the Internet. The RSUs are usually connected to an infrastructured network which is in turn connected to the Internet. Those can communicate directly or via multi-hop executing special applications to send, transmit and receive data in the Ad-hoc domain.

- **Infrastructured domain:** RSUs and *hot-spots* are two access types. The RSUs allow OBUs to access the infrastructure which is connected to the Internet. The OBU can also access the Internet via public, commercial or private Wi-Fi *hot-spots*. If a vehicle is not in the range of the RSU, the OBU can make use of its communication capabilities of cellular networks (Global System for Mobile communications (GSM), General Packet Radio Service (GPRS), Universal Mobile Telecommunications System (UMTS), Worldwide Interoperability for Microwave Access (WiMAX) and Long Term Evolution (LTE)/4G), if the devices are properly equipped.

2.2.5 Technical Challenges

VANETs have special characteristics and behaviors which bring several challenges to the communication among vehicles. In order to provide services for drivers and passengers, some technical issues need to be addressed. For example, scalability and interoperability are crucial aspects because the mechanisms and protocols used should be scalable to numerous vehicles and interoperable with different wireless technologies. The following subsections discuss the most important challenges [28].

2.2.5.1 Reliable Communication and MAC Protocols

Vehicular networks allow multi-hop communication, which can extend the operator fixed infrastructure, providing this way virtual infrastructure among the moving vehicles. Multi-hop represents a major challenge on the reliability of communication. So, efficient MAC protocols need to be in place while adapting to extremely dynamic environments of VANETs, and considering message priority of certain applications (e.g., accident warnings). Fast association and low latency need to be satisfied in the communication between vehicles in order to guarantee:

- service reliability for safety applications.

- the quality and continuity of service for non-safety applications.

Additionally, MAC protocols should consider reliable communication between different technologies (Wi-Fi and GSM).

2.2.5.2 Routing and Dissemination

VANETs experience fast changes in wireless link connections and have also to deal with different amounts of network density [29]. The density in VANETs depends on the local and time. For example, VANETs on freeways or urban areas probably will form a highly dense network during rush hour traffic, while in sparsely populated rural freeways, it is expected frequent disconnections specially during the night. Additionally, it is expected that VANETs can handle a large range of both safety and leisure applications. Therefore, routing and dissemination algorithms should be efficient and adaptive to VANETs characteristics and applications, allowing different priorities for different types of information. Most of the literature in VANETs has focused on the analysis of routing algorithms to handle the broadcasting problem [22] [30] in highly dense areas, assuming that a typical VANETs is always connected. Since vehicular technologies are recent, the dependence from the infrastructure causes a considerable weak penetration in large scale deployments. It is predictable that, in the future, the penetration rate will increase with a larger number of vehicles with communication technologies ready to communicate with each other, reducing the need of fixed infrastructure.

Dissemination will depend on network density as well as on application type. For example, in safety-related applications, message dissemination should be broadcast-type, assuring the message propagation to a group of vehicles but without causing a broadcast storm. In non-safety applications, the transfers should use unicast or multicast transmissions.

2.2.5.3 Security

In vehicular communication, security is the major challenge regarding future deployment and application of VANETs. Security and privacy have a huge importance, and the development and acceptance of services often rely on them. Due to the growing use of discovery protocols, passengers may use services in foreign networks and cause high security problems for them and other users. It is important to create innovative solutions that guarantee security in the communication between users as well as authorized and secure service access.

To improve the vehicular network access solutions, it should be considered:

- Ad-hoc multi-hop communication and authentication concepts, which not only allow secure communication, but also extend the infrastructure coverage with the minimum cost for the operator.

- distributed-based authentication, where security architectures should provide communication between vehicles and allow different service access.

The optimization of authentication is important for infrastructure-based and infrastructureless communication in order to facilitate the reauthentication process, which may happen during a vehicle's mobility.

Additionally, the behavior of the node is an important issue for the security of communication and service delivery in vehicular networks. It is therefore an important aspect to take into account. As vehicular networks are open and dynamic, node cooperation is another aspect worth considering in order to have successful communication between vehicles.

2.2.5.4 IP Configuration and Mobility Management

The V2I architecture is promising in allowing vehicular Internet access and provision of Internet-related services to passengers. Nevertheless, there are two technical challenges under this issue: Internet Protocol (IP) address configuration and mobility management. These challenges compromise the service quality and the service continuity. VANET's characteristics request an automatic and distributed IP configuration. Up to now there is no standard for IP autoconfiguration in Ad-hoc networks, which can be a complex problem in VANETs. Considerable work has been done by some standardization bodies to try to solve this issue. Moreover, the Internet Engineering Task Force (IETF) is working through the *Autoconf WG* in developing Internet Protocol version 6 (IPv6) solutions for Ad-hoc networks, including vehicular network scenarios. International committees are defining architectures for vehicular communication including an IPv6 stack in their protocol stacks.

In relation to mobility management, this is a determinant problem for non-safety applications where the dissemination of information is not made in broadcast. The absence of mobility management mechanisms affects the commercialization of services in VANETs and loses the advantage of V2I architecture, because all Internet-related services will neither guarantee quality of service nor their continuity.

2.2.6 Routing

Routing in VANETs is one of the most researched issues since most of the networks require multi-hop transmissions using vehicles as relays to reach the destination. These networks are unique due to their properties: the large amount of vehicles that exist in the world, their dynamic and almost random route at every moment, and their quick variations of vehicle density in the network. These properties create hard challenges for routing protocol design [6]:

- **Localized operation**: a node takes routing decisions based only on information locally available in the close neighborhood of that node. With this feature, the protocols are highly desirable for VANETs, because its control overhead can be largely reduced by not requiring nodes to know the topology of other parts of the network.

- **Neighborhood discovery**: it is essential to discover the neighbors. For that, beacon messages are sent with different time periods informing neighborhood nodes about their identifier, position, and other relevant information.

- **Trajectory precomputation**: it creates a route for the packets to flow to the destination, it marks the packets to flow through a certain road and, if the network requests it, the relay nodes should find other paths if the source choice is not available due to network changes.

- **Data forwarding**: many Ad-hoc routing protocols create routing tables containing the next-hop to reach the destination based on a given metric but that could be inefficient in dynamic scenarios. Nodes should be able to choose, given the current neighborhood, the best node to forward the packets at the instant it receives them, based on destination's position.

- **Dealing with network partitions**: the high dynamic properties of VANETs sometimes create partitions within the network. This means that it may eventually happen that a relay node does not have other node to forward the message. DTNs deal with this kind of problems implementing the store, carry and forward mechanisms.

- **Prediction of future events**: some information of mobility, such as position and velocity of vehicles, can be useful to predict future positions.

- **Use of additional information**: vehicles are able to use navigation software and even access external information services, providing information about traffic's state which can be a huge help to routing protocols.

According to [6], the following figure (2.7) shows the classification of routing protocols for VANETs:

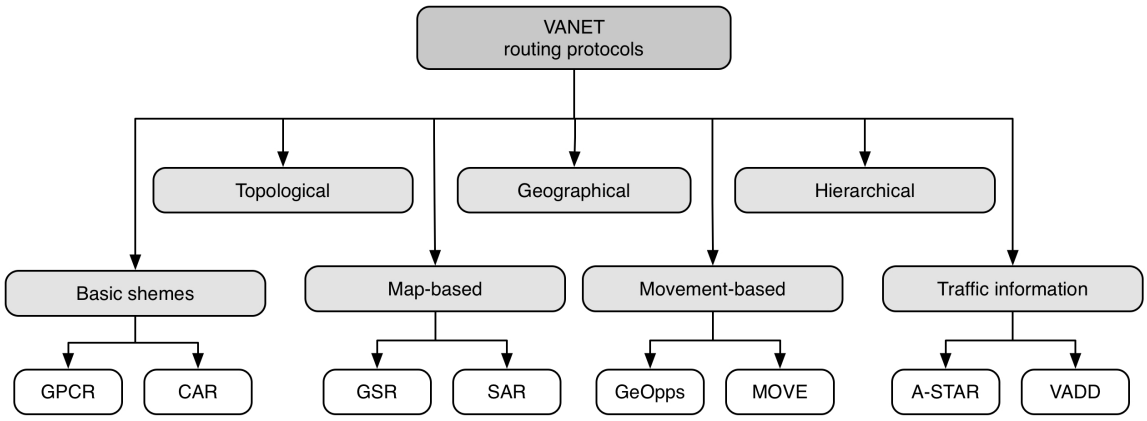


Figure 2.7: Classification of routing protocols

There are also specific protocols for the type of communication: unicast, multicast and broadcast. They can be applied to different values of network density.

2.2.6.1 Topological Routing Protocols

This type of routing protocols uses link information present in the network to deliver data packets from source to destination with nodes that compose the network. Topology based protocols can be proactive by creating tables even when there is no message to route, i.e. they find paths in advance for all source and destination pairs. They can also be reactive, which means that they find a route to the destination only when it is requested, i.e., when there is data to be transmitted.

According to [31] and [32], proactive routing protocols have low latency for real-time applications and do not need to discover routes, but the unused paths waste a considerable part of bandwidth. The reactive ones save bandwidth because they are beaconless and there is no need to flood the network on every routing table's update; however, they take more time to react to route changes.

2.2.6.2 Geographical Routing Protocols

The geographical based routing protocols depend on the positioning data information given by GPS receivers on each node [33]. While in topology based protocols the addressing specifies a single node, in this type of protocols the addressing specifies an area and each node knows its own and neighbors' positions. These protocols neither need to maintain routing tables nor need route discovery. The decisions are made based on GPS information.

These protocols are highly appropriated for high mobility environments. The drawbacks are related to privacy (by getting the constant position of people) and to the operation of GPS in closed spaces, like underground parks or tunnels.

2.2.6.3 Hierarchical Routing Protocols

Hierarchical routing protocols [34] are used to optimize the resource usage and efficiently maintain the energy consumption of sensor nodes, by involving them in multi-hop communication within a particular cluster (set of nodes connected for a defined period of time), and by performing data aggregation and fusion in order to decrease the number of transmitted messages. The messages are sent from cluster to cluster by using gateway nodes.

This type of routing has also some drawbacks such as the increase of overhead needed to build the clusters. So, the use of this routing solution should not depend only on clusters, because in some cases they might be composed by a small number or they might even not exist. Other routing solutions should be implemented in order to make use of clusters when it is possible and suitable, by limiting the number of re-transmitters and increasing the performance of routing strategies [35].

2.2.6.4 Movement-based Routing Protocols

These protocols use a street map, considering the movement conditions of the intermediate nodes, as well as the source and the destination nodes. They know about the speed and direction of the nodes involved in the routes and can predict their positions in a near future and, eventually, by knowing the amount of data to send, the algorithm can know how long the transmission will take [36].

2.2.6.5 Basic Schemes

These protocols work without information specific for VANET scenarios, such as a map of the city, statistics about traffic density on the different roads, speed limits, information about trajectory estimations. They are able to work only with control messages from neighbors. Two protocols that follow this kind of routing are CAR [37] and GPCR [38]. The first saves the coordinates of the points where the direction of the packets sent has changed, while the second includes special roles for nodes depending on their location.

2.2.6.6 Based-on Trajectory

These routing protocols work with information about the planned trajectory of the vehicle. Navigation systems are currently used and are able to suggest routes to the drivers, so not only the current trajectory but also the whole route up to a given destination can the estimated. With this information, trajectory-based protocols forward the packets to those neighbors that are expected to get closer or quickly to the destination.

For example, in GeOpps [39] a node sends a message to a neighbor if its trajectory passes closer to the destination than the current node. MOVE [40] regards to velocity of the nodes to select the most adequate next node.

2.2.6.7 Traffic Information

Traffic Information routing protocols assume that vehicles can obtain information about the traffic conditions. For example, A-STAR [41] knows the density of vehicles at each street and takes it into account when choosing the best path from the source to the destination. VADD [42] uses the same information to decide if a message should be sent immediately or stored until more connected street.

2.3 Delay Tolerant Networks

2.3.1 Definition

Delay and Disruption Tolerant Networks (known by the abbreviation DTN) [8] [43] [18] are networks that have the potential to establish communication where connectivity is sparse and intermittent, where long and variable delay, high latency, high error rates, highly asymmetric data rate and even no end-to-end connectivity exist.

The Delay-Tolerant Networking concept was initially proposed as an approach for InterPlanetary Internet (IPN) [44][45]. Deep space communication is characterized by very large latencies, low data rates, possibly time-disjoint periods of reception and transmission, and intermittent scheduled connectivity. Sometimes, due to their trajectory, the satellites pass through a shadow area from the emitter. That means that they are not in direct contact with the receptors, and to not lose information while they are in those areas, they store the information they are receiving. This way there is no loss of information during that period.

DTNs allow to interconnect devices in regions where current networking technology cannot reach. The concept is that an end-to-end connection may never be present. To communicate, intermediate nodes take custody of data being transferred and forwarded it when there is an opportunity. The connections and nodes can be uncertain and disconnections can be long-lived.

Figure 2.8 shows an example of the concept of DTN in the daily routine, that can be used to provide access to the Internet in rural areas. The laptops are communicating with each other in a remote area, and to communicate with the Internet, the village can be serviced via a router attached to a bus traveling between the rural area and the Internet gateway. The requests from the laptops are delivered to the gateway by the mobile device, and then it provides the responses on its next trip.

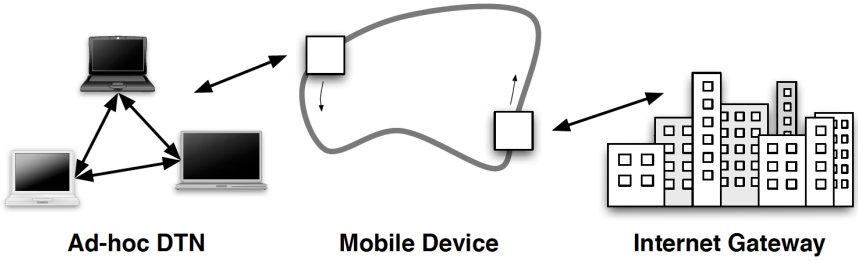


Figure 2.8: Group of laptops communicating with each other and the Internet via DTN [7]

2.3.2 Characteristics

The main aim of DTN is to solve the problem caused by networks with disruptions and large delays. The most common TCP/IP networks provide connection oriented services. According to this protocol, the use of the Internet depends on some important assumptions [9][45][46]:

- Continuous and bidirectional end-to-end path: A continuously available bidirectional connection between the source and the destination is the solution to provide applications with a reliable data transfer.
- Short Round Trip time: Small delay sending data packets plus the time it takes for an acknowledgment confirming the received data.
- Symmetric Data Rates: Relatively consistent data rates between the source and destination, in both directions.
- Low Error Rates: Relatively little loss and small signal propagation latencies.

However, there are several scenarios where these assumptions cannot be satisfied. The characteristics of DTNs are markedly different from the common networks. These networks are characterized by:

- Intermittent Connectivity: With no end-to-end path between source and destination using the Transmission Control Protocol (TCP), protocols do not work. Other protocols are required.
- Long and Variable Delay: Intermittent connectivity, long propagation delays between nodes, and variable queuing delays at nodes contribute to end-to-end path delays that can not be accepted in the Internet, or applications that rely on quick return of acknowledgments or data.
- Asymmetric Data Rates: Some protocols do not allow large assymetric data rates.
- High Error Rates: Bit errors on links require correction.

In a common TCP session, timeouts caused by long delays cause the termination of the session, as well as, when a node or a link unavailable for a long time, it produces data losses or session disruption. DTN protocols provide robustness, being able to work in the environments mentioned above by introducing tolerance for delays and intermittent connectivity. In theory, DTNs can handle nodes or links unavailable for several days whilst still being able to provide reliable data transfer due to buffering capacity in the network.

The DTN protocol implementations and some specifications for them are characterized by two main documents: "Delay-Tolerant Networking Architecture" [47] and "Bundle Protocol Implementation" [48]. Some concepts in these documents will be described and analyzed in the next subsections.

DTN can use multiple delivery protocols including TCP/IP, raw Ethernet, serial lines or hand-carried storage drives for delivery. Each of those protocols provide different semantics. A collection of protocol-specific Convergence Layer Adapters (CLA) provide all the necessary functions to carry DTN protocols.

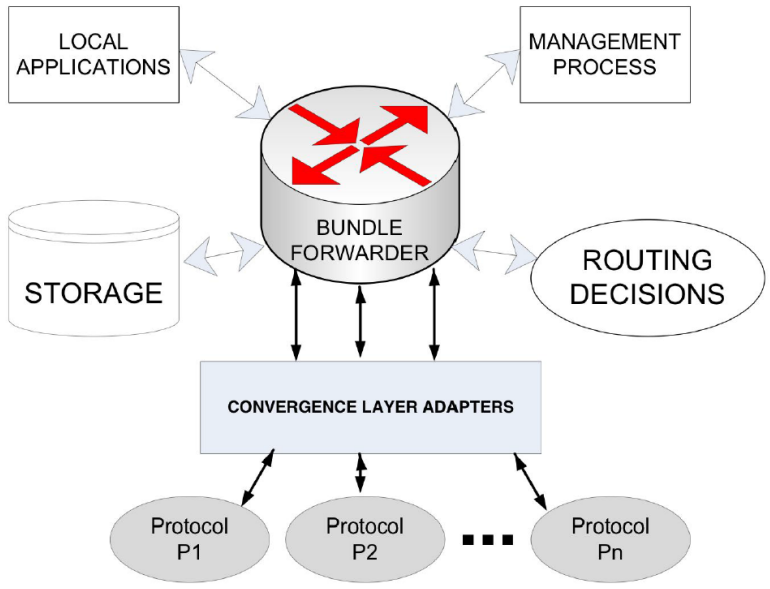


Figure 2.9: Bundle Forwarder Interaction Architecture [8].

As it is shown in figure 2.9, the central unit is the blundle forwarder which is responsible for moving bundles between applications, storage and CLAs according to decisions made by routing algorithms. The arrows represent interfaces which carry either bundles or directives (applications, CLAs, routing decisions or management messages).

2.3.3 Architecture

The DTN architecture implements receive, store and forward mechanisms that allow the communication from source to destination when there is no end-to-end path. The nodes are able to overcome the problems related to intermittent connectivity and long/variable delays, by using the concept of Store and Forward message switching shown in the figure 2.10. Just in case the network fails during transmission, each node

has a persistent storage system which is used as a backup. This storage can be made for minutes, hours or even days and the nodes need to be equipped with hard disks or flash memories. The blocks of data or bundles are forwarded from a node to another along a path, such that the message eventually reaches the destination.

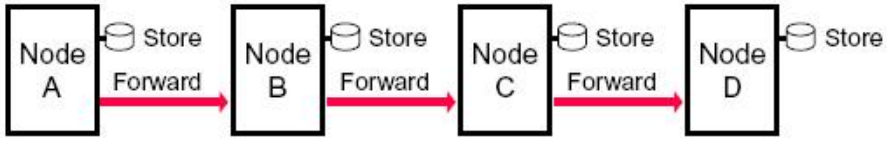


Figure 2.10: Store and Forward Message Switching [9].

Persistent storage is needed in this type of networks for several reasons [9]: a communication link may not be established between one node and another for a long period of time; one node in a communication pair can send or receive data much faster than another and, when a message is being transmitted, if an error occurs during the transmission, a message may need to be re-transmitted.

DTNs implement Store and Forward mechanism by overlaying a new protocol layer called "Bundle Layer" on the top of lower layers, as is it shown in the figure 2.11. The bundle layer links the region specific lower layers so that the application programs can communicate across multiple regions. Bundles are also called messages and consist in: the source application's user data, control information provided by the source application for the destination application which describes how to process, store, etc. Bundle has a header which is added by the bundle layer, and can be as large as the volume of the data encapsulated.

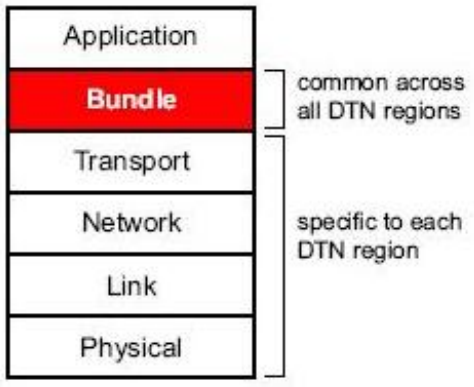


Figure 2.11: DTN layers [9]

The bundle layer stores and forwards the entire bundles in the DTN network. A single bundle layer is used across all DTN regions in the network. The figure 2.12 shows the DTN bundle layer, common across all DTN regions while the lower layers (transport layer and below) are chosen for their appropriateness to each region. This layer communicates between nodes on intermittent networks using simple sessions with minimal round trip times. The acknowledgments are an optional service.

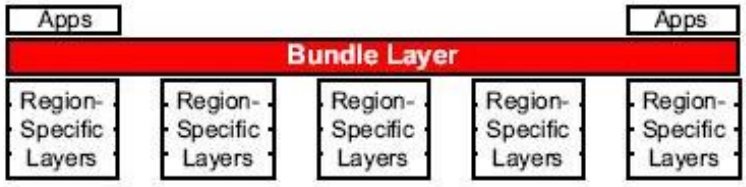


Figure 2.12: Common bundle layer accros all DTN regions [9]

2.3.3.1 DTN Nodes

In DTN, the nodes are entities with a *bundle layer* attached, and the node may work as a host, router or gateway or as a combination which represents the source, relay and destination [9].

- **Host:** sends and/or receives bundles. A host can be a source or destination of a bundle. The bundle layers of hosts require persistent storage in which they queue the bundles due to its operation over a long-delay.
- **Router:** forwards bundles within a single DTN region and may optionally be a host. In the routers, the bundle layer operates over long delay links and requires persistent storage, where the bundles are queued until another node is available to forward the bundle.
- **Gateway:** forwards bundles between DTN regions and may optionally be a host. Gateways must have persistent storage too.

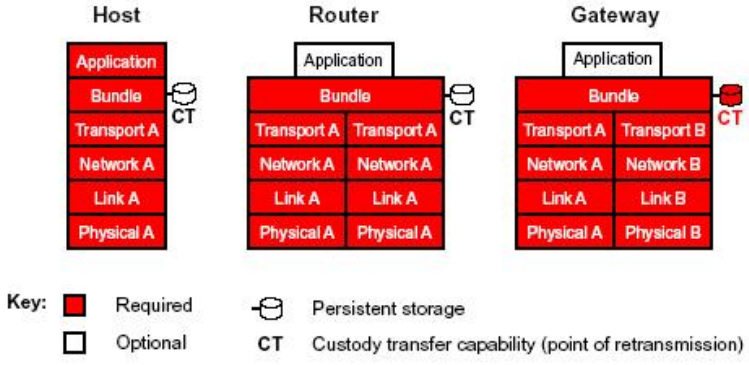


Figure 2.13: Different types of DTN nodes [9]

2.3.3.2 Naming, Addressing and Binding

Fall *et al.* [8] considered naming and addressing the most important aspects of a network architecture, as well as one of the most challenging. Names have generally variable-length strings, while addresses are expected to be fixed-length identifiers. Mapping and binding are used to convert names into addresses. In case of the Internet,

Domain Name System (DNS) is responsible for this function, while in overlay network system, it can be done in a locally-executed hash function.

During the evolution of DTN, nodes have had identifiers which are used in the context of bundle protocol [48]. This protocol provides basic message delivery service for DTN. Originally, identifiers in bundle protocol were built considering three parameters: region, host and application which was able not only to identify a host, but also an application of interest on the host. A region was a portion of the network topology and was assumed to represent a well-connected area surrounding a planet (in the original IPN design).

After some considerations, the region construct was considerably modified because of the mobility of the nodes, and they may have multiple network interfaces. So, additional flexibility was required in how they were named, and so multiple namespaces were supported with different naming semantics. This property allows hosts to have multiple identifiers and even multiple types of identifiers. In order to specify nodes with multiple identifiers, IETF worked in the area of generalized naming systems, in the form of Universal Resource Identifiers (URI)s. URIs have a few important characteristics:

- Allocated Name Spaces: a URI has the form <scheme>:<scheme-specific-part>, where the scheme part is a string allocated from a set of several scheme names as http, file, sip, etc.
- Variable Length: Although limited to a relatively large size, URIs are essentially free-form.
- Structured Semantics: URIs have to obey to a general syntax and semantics, but a new scheme may define its own special additional semantics.

In DTNs, URIs are referred to Endpoint Identifiers (EID). For messages containing DTNs, URIs with symbolic names, some *binding* step is performed by one or more node along the path to the destination. When DNS is used at the sending node, it is called a case on *early binding*, where a DNS name is linked to an IP address. Since DTN supports direct forwarding based on symbolic names, the *early binding* used by DNS in the Internet is not required. With this property, messages are passed along nodes towards their destinations based on forwarding entries present at DTN routing nodes. This characteristic is known in the DTN literature as *late binding*.

2.3.3.3 Bundle blocks and Primary Bundle Fields

In the DTN architecture, the applications operate with bundles, which are messages carried with variable-length protocol, Protocol Data Unit (PDU). The bundle block contains the basic information needed to route bundles until their destination.

The bundle block shown in figure 2.14 has several fields:

- **Bundle Processing Control Flags** is a Self-Delimiting Numeric Value (SDNV) that contains the bundle processing control flags (indicating if the bundle is a fragment, acknowledgment, a request of custody or other parameters).
- **Block Length** is a SDNV that contains the aggregate length of all remaining fields of the block.

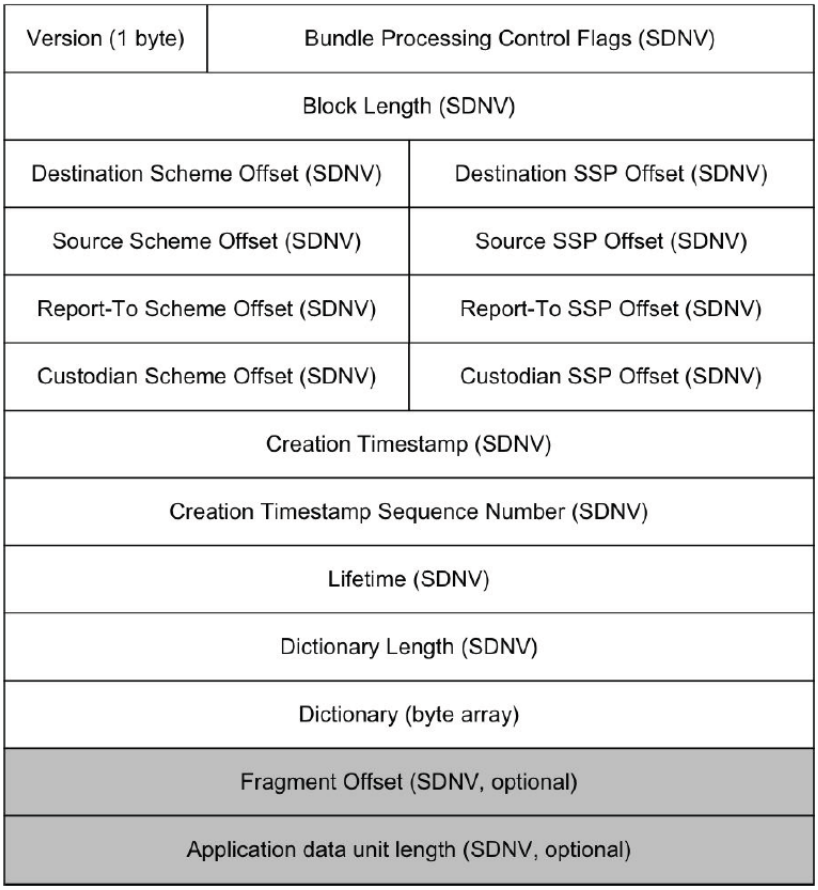


Figure 2.14: Bundle Block Fields [9].

- **Destination EID** is the EID of the destination, i.e., the node at which the bundle is to be delivered.
- **Source EID** is the EID of the source, i.e., the node from which the bundle was initially transmitted.
- **Report-to EID** identifies the node where reports shall be sent. It may identify the EID of the source.
- **Custodian EID** identifies an EID whose membership includes the node that most recently accepted custody of the bundle upon forwarding this bundle.
- **Creation Timestamp** is the concatenation of the bundle's creation and an increasing sequence number in order to guarantee its oneness for each Application Data Unit (ADU) generated at the same source. Bundle creation time is expressed in seconds since the start of the year 2000, on Coordinated Universal Time (UTC), time at which the transmission request was received which resulted in the creation of the bundle. This means that DTN nodes shall have time synchronization capability.
- **Lifetime** is an SDNV that indicates the time at which the bundle's payload will

no longer be useful, encoded as a number of seconds past the creation time. If the current time is larger than the creation time plus lifetime, bundle nodes need no longer retain or forward the bundle. The bundle may be deleted from the network.

- **Dictionary Length** is an SDNV that contains the length of the dictionary byte array.
- **Fragment Offset** is an SDNV field that indicates the offset from the start of original application data unit at which the bytes comprising the payload of this bundle were located (just in case of the Bundle Processing Control Flags indicate that the bundle is a fragment).
- **Total ADU Length** indicates the total length of the original application data unit of which this bundle's payload is a part (just in case of the Bundle Processing Control Flags indicate that the bundle is a fragment: if not, the Total ADU Length field is ommited from the block).

2.3.3.4 Bundle Fragmentation

It may be necessary for bundle protocol agents to reduce the sizes of bundles in order to forward them. In case of the node to which a bundle is to be forwarded is accessible only via intermittent contacts and the bundle is big enough to not be transmitted for only one time, the size of the bundle needs to be reduced by "fragmenting" the bundle.

According to [47], fragmentation can be classified in two different forms:

- **Proactive Fragmentation:** it consists in a DTN node dividing a block of application data into multiple smaller blocks, and transmitting each block as an independent bundle, i.e., in this case, each block is transmitted individually by the source. The destination node is responsible to extract the smaller blocks from incoming bundles and reassemble them into the original entire block. This method is called proactive fragmentation because it is used before the transmission when contacts are known in advance.
- **Reactive Fragmentation:** it is the process of fragmenting the bundle when only a part of the entire bundle is transferred. In this case, the receiving bundle layer modifies the bundle fields to indicate it is a fragment and then forwards it normally. After sending a portion of the bundle to the next-hop, the next portions will be sent when subsequent contacts become available (it can be sent via different next-hops.)

In DTN, fragmentation and reassembly are designed to improve the efficiency of bundle transfers by ensuring that all contacts are fully used, and by avoiding the retransmission of partially-forwarded bundles.

2.3.3.5 Custody Transfer

The node receiving the bundle during its path to the destination is called "custodian", as long as it agrees to accept the reliable delivery responsibility. In DTN,

custody transfer is an optional service provided to a bundle [8]. This service consists of keeping track of a current "responsible entity" for each bundle, and the "custodian" is required to keep the bundle safe in persistent memory until another custodian has received it successfully.

A bundle custodian must store a bundle until either:

1. Another node accepts the bundle
2. Expiration of the bundle's Time-to-Live (TTL), which is aimed to be much longer than the time a custodian takes to send the Acknowledgement (ACK).

The bundles may be moved from one custodian to another, and an acknowledged transfer is performed for each bundle. This mechanism allows the source to pass the responsibility and recover its re-transmissions-related resources, after sending the bundle.

This process provides a method for tracking bundles, information about the nodes that participated in the transfer, as well as it provides a mechanism for enhancing the reliability of message delivery.

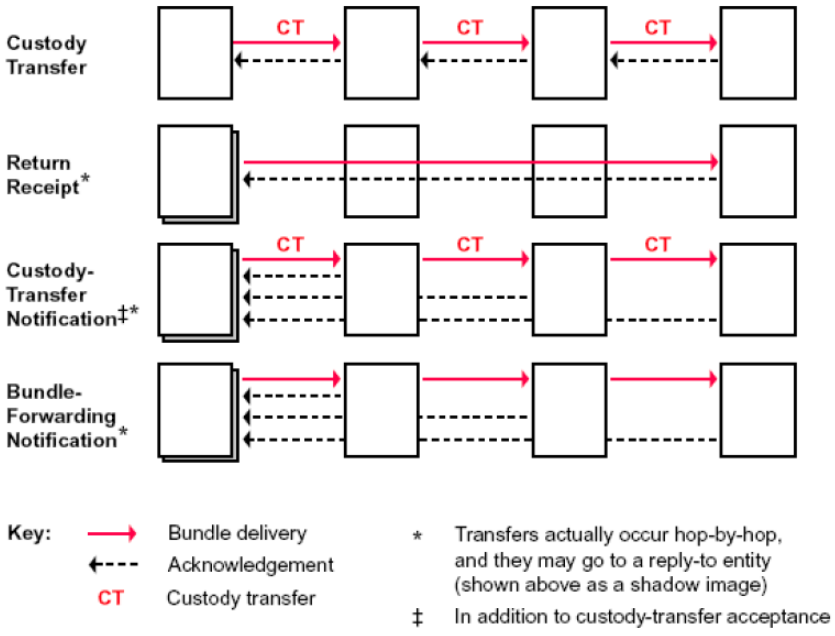


Figure 2.15: Custody transfer and class of service [10]

The figure 2.15 presents Classes of Service (CoS) for a bundle provided by bundle layer [10]:

- **Custody transfer:** Delegates the transmission responsibility to an accepting node, which returns a custodial-acceptance acknowledgment to the previous custodian.
- **Return Receipt:** Destination sends an acknowledgement confirming that the bundle has been received by the destination application.

- **Custody-Transfer Notification:** Each node that receives the custody sends a notification to the source, when a node accepts a custody transfer of the bundle.
- **Bundle-Forwarding Notification:** Notification to the source, whenever the bundle is forwarded to another bundle.
- **Authentication:** The method used to verify the sender's identity and the integrity of the message.

2.3.3.6 Security

Most of networks security methods mutually authenticate user identities and the integrity of messages, but they do not authenticate the routers involved in the communication and the forwarded information. In DTN, forwarding nodes (routers and gateways) must be authenticated and information sent from the source should be authenticated by forwarding nodes, so that the network resources can prevent the carriage of prohibited traffic. In some cases, it is not acceptable for an unauthorized user to flood the network with traffic, denying service to authorized users. Another case is when it is also not possible for unauthorized traffic to forwarded over certain network links like mission-critical links.

There were established several goals for the security component of the DTN architecture [47]:

- Deny unauthorized applications from having their data carried through or stored in the DTN.
- Deny unauthorized applications from declaring control over the DTN infrastructure.
- Prevent authorized applications from sending bundles at a rate or CoS for which are not allowed.
- Discard bundles that are damaged or modified in transit.
- Detect and deny compromised entities.

Many access control and authentication protocols designed for operation in low-delay and connected environments may not work well in DTNs due to their properties. Approaches that require frequent access to centralized servers to complete the authentication are not attractive.

To satisfy these security requirements, the DTN architecture adopts a standard architecture, DTNSEC [47], which uses hop-by-hop and end-to-end authentication as integrity mechanisms. The use of both approaches is to be able to handle access control for data forwarding and storage apart from application-layer data integrity. The hop-by hop mechanism is aimed to authenticate DTN nodes as legitimate transceivers of bundles between them, while the end-to-end mechanism provides authentication for a principal such as a user.

In light of the goals above, DTN nodes discard traffic as early as possible if authentication or access control checks fail.

2.3.3.7 Scheduled and Opportunistic Contacts

As was previously referred, in DTN environments, most of the times the direct contact between source and destination is not available. So, different types of scheduled and opportunistic contacts exist:

- **Persistent contacts** are used between nodes that have persistent network connection, for instance, the RSUs.
- **On-demand contacts** are used for nodes that can establish a connection when needed, for example, dial-up connections.
- **Intermittent - Scheduled contacts** are used for nodes that are available for communication at certain and pre-determinated times, for example, orbiting satellite.
- **Intermittent - Opportunistic contacts** are used for irregular connection where there is no guarantee that the contact will be available.
- **Predicted contacts** are used as a hybrid of scheduled and opportunistic contacts where a future connection is predicted based on a mobile node's movement pattern, for example, public buses.

2.3.4 Routing

In DTNs the forwarding of messages is not easy when compared to traditional networks, where most of the times there is an end-to-end path. These networks are characterized by being tolerant to the lack of connectivity between end-to-end paths. In those scenarios, the routing protocols often used in Ad-hoc networks do not provide an infrastructure.

One of the most important feature of a DTN is the contact scheduling, which can be approximated to a spectrum of density that shows the predictability of encountering each other in different scenarios, as it is shown in the figure 2.16.

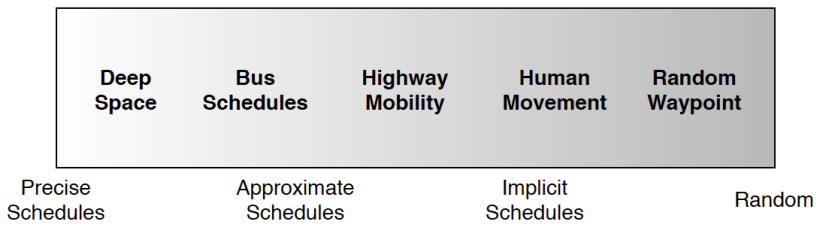


Figure 2.16: Spectrum of contact schedule predictability [7]

These networks have some specific properties that should be taken into consideration when designing a routing protocol:

- **Contact Schedules** depend strongly on the application area under consideration.

- **Contact Capacity** is the amount of data that can be exchanged between two nodes during a contact. This depends on the link technology and on the duration of the contact.

- **Buffer Space** is where the bundles can be buffered for long periods of time (hours, days) in case of long disconnections. This means that the intermediate routers that forward the messages require enough buffer space to store all the messages that are waiting for a future communication with the next node. An alternative to have large amounts of space to storage is that routing strategies might need to consider the available buffer space when they are deciding the next-hop.

- **Processing Power**, for example, in the case of sensor networks, their processing capability is very small (in terms of CPU), so they may not be able to run complex routing protocols compromising this way routing strategies.

- **Energy** can be limited, either about the mobility of the nodes or just because they are located in places where the connections to power supply are difficult. Routing consumes energy by sending, receiving and storing messages. Researchers have investigated general techniques for saving power in DTNs [49].

To achieve the DTN vision [7], routes must be found over uncertain hops and intermittently connected nodes.

When a message arrives at a node, it might not exist an available path until the final destination. The node needs to store the messages and, when a contact is established with other node, the node has to decide if it forwards the message or not. This decision is a delicate issue, since it depends on several conditions. In some circumstances, the best way to deliver the messages is to set a fixed threshold, and the messages are only sent if the delivery probability between two nodes in contact is greater than a certain value. Besides, if two nodes are in contact and they have a low delivery probability, it is not certain that transferring the message between the two nodes is the right option to take, because it can take a lot of time to find another node with higher probability. This way, routing in DTNs is constantly dynamic and it is not known which is the right option to take.

Thus, forwarding the message to a large number of nodes will certainly increase the chance of delivering it to the destination, but it will also consume a lot of resources to store and process all the replicas. On the other hand, forwarding the message to a small number of nodes will save resources, but it will decrease the chances of delivering it to the destination. In order to have a good relation between the use of resources and the delivery ratio, some considerations need to be taken into account.

The routing protocols in DTN can be classified in two different strategies [7]: flooding and forwarding replication. The flooding ones are replication-based and describe how a routing strategy depends on multiple copies of each message. The forwarding strategy is based on the knowledge and describes how information about the network is used to make decisions.

There are advantages and disadvantages in both approaches, which are heavily dependent upon the nature of the scenario at hand. Forwarding-based approaches are generally much less wasteful of network resources, as only a single copy of a message

exists in storage in the network at any given time. Furthermore, when the destination receives the message, no other node can have a copy. This eliminates the need for the destination to provide feedback to the network to indicate that outstanding copies can be deleted. However, statistics show that forwarding-based approaches do not allow for sufficient message delivery rates in many DTNs. Replication-based protocols, on the other hand, allow for greater message delivery rates, since multiple copies exist in the network, and only one must reach the destination. However, the trade-off here is that these protocols can waste valuable network resources. Furthermore, many flooding-based protocols are inherently not scalable.

The most common routing protocols used in DTN scenarios are the following:

2.3.4.1 MaxProp

MaxProp [50] is a flooding-based protocol, but it falls into the category of replication based protocols. This protocol uses several mechanisms working together to increase the delivery rate and decrease the latency of delivered bundles. In this protocol, a node floods the messages but explicitly clears them once a copy gets delivered to the destination [18]. This protocol also incorporates "intelligence" by determining which messages should be transmitted first and which messages should be dropped first. MaxProp maintains an ordered-queue based on the destination of each message; it sends messages to other hosts in a specific order that takes into account message hop counts and message delivery probabilities based on previous encounters. For instance, when two nodes meet, they first exchange their estimated node-meeting probability. Ideally, every node will have a probability to meet every other node. With these pieces of information, the node can then compute a shortest path until the destination node.

2.3.4.2 RAPID

Resource Allocation Protocol for Intentional DTN (RAPID) [51] was developed at the University of Massachusetts, Amherst. It is also a Replication Based Flooding Protocol. RAPID can optimize a specific routing metric, such as worst-case delivery latency or the fraction of packets that are delivered within a deadline. The main goal of this protocol is to intentionally effect a signal routing metric.

This protocol models DTN routing as an utility-driven resource allocation problem that translates the routing metric into per-packet utilities, which determine at every transfer opportunity if the marginal utility of replicating a packet justifies the resources used.

According to [52], the main aim of RAPID is based upon an utility function that assigns an utility value, U, which is based on the metric being optimized like end-to-end delay or packet delivery ratio. RAPID replicates packets in decreasing order of their marginal utility at each transfer. The protocol works in four steps:

- Initialization: Metadata is exchanged to help estimating packet utilities.
- Direct Delivery: Packets destined for immediate neighbors are transmitted.
- Replication: Packets are replicated based on marginal utility.

- Termination: The protocol ends when contacts break or all packets have been replicated.

2.3.4.3 Epidemic

Epidemic [12][18] routing is a flooding-based routing protocol. This mechanism guarantees that all nodes will eventually receive the messages. Epidemic replicates and transmits messages to all encountered peers that still do not have them. If contacts between nodes are long enough and the message storage space is unlimited, epidemic minimizes the delivery delay and maximizes the delivery ratio.

This routing protocol is based on the theory of epidemic algorithm, exchanging messages between nodes when they get in contact with each other. When two nodes meet, they exchange summary vectors which is an index of the messages. After that, each node can determine if the other node that is in contact with itself has some new message. In that case, the node requests the message, as it is shown in the figure 2.17.

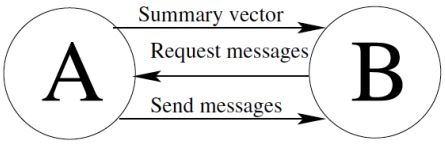


Figure 2.17: Epidemic routing message exchange [11].

The main goals of epidemic routing are the following [12]: i) to distribute efficiently the messages through partially connected Ad-hoc networks in a probabilistic way; ii) to reduce the amount of resources consumed in delivering any single message; and iii) to maximize the percentage of messages that are delivered to their destination.

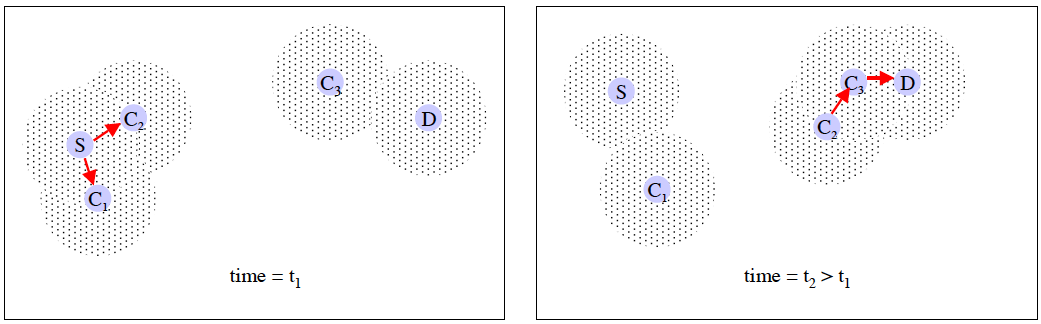


Figure 2.18: Epidemic mechanism [12].

The figure 2.18 shows an example of Epidemic routing at high level. The mobile nodes are represented as circles, and their wireless communication range is represented as a dotted circle from the center of the node. In the left figure, the source, S, wants to send a message to the destination, D, but there is not an end-to-end connection between S and D. S replicates its message to its neighbors, C1 and C2, within direct

communication range. Later, as it is shown in the figure on the right, C2 moved until where it established a direct communication range with another node, C3, and replicates the message to it. C3 has direct contact with D and transmits the message to the final destination.

Since all devices are normally limited, epidemic wastes storage and bandwidth in comparison with other protocols. However, under some circumstances, it may be the only viable technique for successfully delivering application data.

2.3.4.4 Spray and Wait

The Spray and Wait protocol [53], developed by researchers at the University of Southern California, is a routing protocol that attempts to gain delivery ratio benefits from the replication based routing and low resource consumption benefits of forwarding-based routing.

Spray and wait consists in two phases: the spray phase and the wait phase. This mechanism generates L copies of the message, that is the maximum allowable copies of the message in the network. During the spray phase, the source of the message is responsible for spraying, or delivering, one copy to L distinct nodes (relays). When a relay receives the copy, it begins the wait phase, where the relay simply holds the message until the destination is directly encountered.

2.3.4.5 PRoPHET

Although MANETs are considered moving completely random, in fact they move in a predictable fashion based on repeating behavioral patterns. If a node has visited a specific location several times, it is likely that it will visit the same location again. Probabilistic ROuting Protocol using History of Encounters and Transitivity (PRoPHET), makes uses of these considerations and information to improve routing performance.

A probabilistic metric was establish, called *delivery predictability*, $P_{(a,b)} \in [0,1]$, at every node a for each known destination, b, that indicates how this node will be able to deliver a message to that destination. PRoPHET's operation is quite similar to Epidemic's (Figure 2.17): when two nodes meet, they exchange summary vectors that contain the delivery predictability information stored at the nodes. This information is used to update the internal delivery predictability vector, and then the summary vector is used to decide which messages are going to be exchanged between two nodes based on the forwarding strategy used.

The calculation of delivery predictability is composed by three stages:

- The first stage is the metric update whenever a node is encountered, increasing the delivery predictability. The calculation is shown in the equation 2.1, where P_{init} is an initialization constant.

$$P_{(a,b)} = P_{(a,b)old} + (1 - P_{(a,b)old}) \times P_{init} \tag{2.1}$$

- The second stage is when a pair of nodes does not meet the other in a while, which means that the probability for them to deliver a message to each other is lower.

Thus, the delivery predictability must *age* as it is shown in the equation 2.2, where γ^k is the *aging constant* and k is the number of time units that have elapsed since the last time the metric was aged. The time unit should be defined based on the application and the expected delays in the network.

$$P_{(a,b)} = P_{(a,b)old} \times \gamma^k \tag{2.2}$$

- The third stage is relative to a *transitive* property that is based on the frequency that node A encounters node B, and the frequency that node B, encounters node C, then node C probably is a good node to forward messages that have as destination node A. Equation 2.3 shows how the delivery predictability is affected by the *transitivity*, where $\beta \in [0, 1]$ is a scaling constant that decides how delivery predictability is influenced by the impact transitivity.

$$P_{(a,c)} = P_{(a,c)old} + (1 - P_{(a,c)old}) \times P_{(a,b)} \times P_{(a,c)} \times \beta \tag{2.3}$$

To better understand the mechanism of PRoPHET routing, figure 2.19 presents an example of the transitive property of the delivery predictability and the basic operation of PRoPHET. In the figure it is shown the scenario where node A has a message and wants to send it to node D. In the subfigures a) to c), the delivery predictability tables for the nodes are shown. Let's assume that nodes C and D encounter each other frequently in subfigure a), increasing the delivery predictability values they have for each other to *high*. Then, assume that node C also frequently encounters node B and the delivery predictability values they have for each other are also *high* in c). This will mean that the transitive property will also increase the value B has for D to a medium level. Finally, node B encounters node A that has a message for node D. The subfigure d) shows the message exchange between node A and node B. They exchange summary vectors and delivery predictability information and, as a consequence, delivery predictabilities are updated, and node A then realizes that $P_{(b,d)} > P_{(a,d)}$ and thus forwards the message for D to node B.

Lindgren *et al.* [11] analysed the performance of PRoPHET and epidemic protocols in two different scenarios: in a random mobility scenario and in a community scenario, where the nodes go regularly to specific places.

In the results they concluded that, in the random mobility scenario, the performance is similar for both protocols with slightly better results for PRoPHET specially with short communication range and high number of hops. The delivery ratio and delay in PRoPHET have lower communication overhead so being more efficient. This is related with the forwarding of messages in PRoPHET, since it only forwards to the nodes with large probability to encounter the destination, while epidemic routing sends the messages to every node that enters in contact. The results also showed that, in a completely random mobility scenario, PRoPHET still operated in a better way than epidemic.

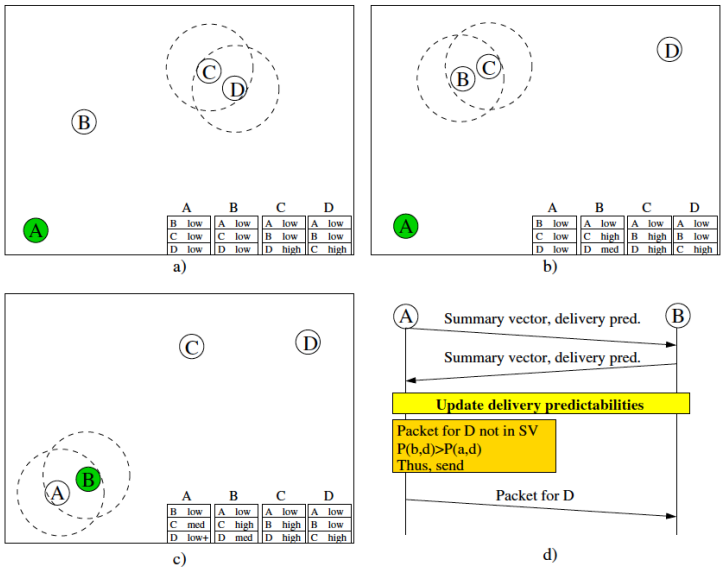


Figure 2.19: PRoPHET example [11].

2.4 Vehicular Delay Tolerant Networks

VANETs have some problems related with the speed and mobility of vehicles causing short duration contacts. Also the existence of buildings, tunnels and other infrastructures compose radio obstacles leading to disruption and intermittent connectivity. The main difference between VANETs and VDTNs is that in VANETs it is assumed an end-to-end connectivity while in VDTNs it is not [54][55][56].

VDTN is a specific case of DTN that gives support to vehicular communications made in very adverse conditions, where the solutions used to establish a point-to-point connection can not guarantee the communication. Adverse conditions are those where there is frequent loss of connectivity, large latency or delays. These networks are also used to provide communication tools to remote areas, as villages and mountains, where there are no infrastructures for communication.

VDTNs may be defined as Ad-hoc networks without a predefined organization, where the vehicles are equipped with communication devices cooperating between them (and with an existing infrastructure), in order to enable the transmission of messages that otherwise would be difficult or even impossible to transmit.

2.4.1 Applications

There are several examples of VDTN applications:

- **improve road safety** by providing an advisory/warning to the driver about the conditions of the road, cooperative collision avoidance and emergency break warning (as in the example shown in figure 2.20).
- **optimize the traffic flow**, preventing road congestion, informing passengers about the traffic conditions or adapting an intelligent speed.

- **sensor networks** collecting data from outside weather conditions (meteorological, noise, air quality and brightness sensors) parking sensors, level sensors in the garbage bins and road surface conditions.
- **commercial applications** marketing data, touristic information, parking space availability, advertisements, etc.
- **provide connectivity to remote areas** providing access to the mail boxes or simple transfer of files.

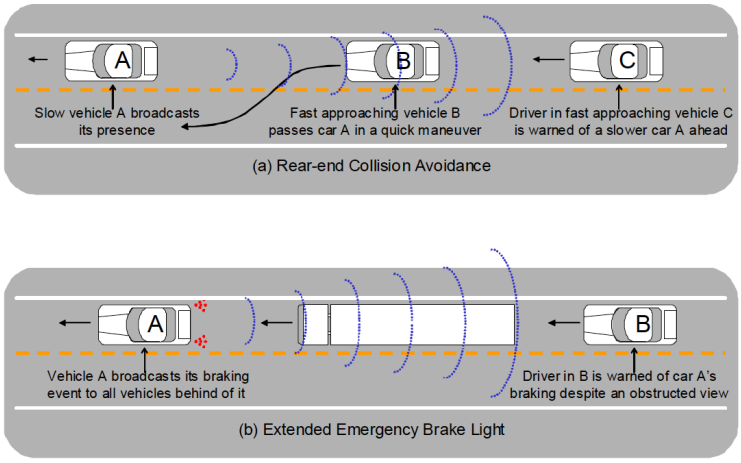


Figure 2.20: Vehicle safety communication examples [4].

According to [13], [57], [58] and [59], applications in VDTNs are classified in four different types according to figure 2.21):

- The first type is general information services, where delayed or lost information does not compromise safety or render application useless. Examples of communicated information in this type are weather reports, web browsing, business services, road conditions, traffic volume and in context-specific broadcasts (leisure feeds and advertising).
- The second is about information services for vehicular safety, which delayed information may result in compromised safety or render application useless. As examples of communicated information are safety alerts associated to potential dangers, such as abnormal surface conditions or vehicle behavior.
- The third is about individual motion control using inter-vehicle communication. Applications that send warnings to the driver or regulate local vehicle actuators to guarantee safe and efficient operation. As examples are the adaptive cruise-control and other improvements to avoid vehicle collisions.
- The fourth is about group motion control using inter-vehicle communication. Vehicle motion planning involving global optimization that may or may not involve group motion regulation controlled by centralized or distributed applications.

However, in these applications, vehicles are commonly organized into groups to facilitate complementary trajectory planning and have the possibility to couple their motion to one another. Explicit membership may be used to identify participants and establish relationships.

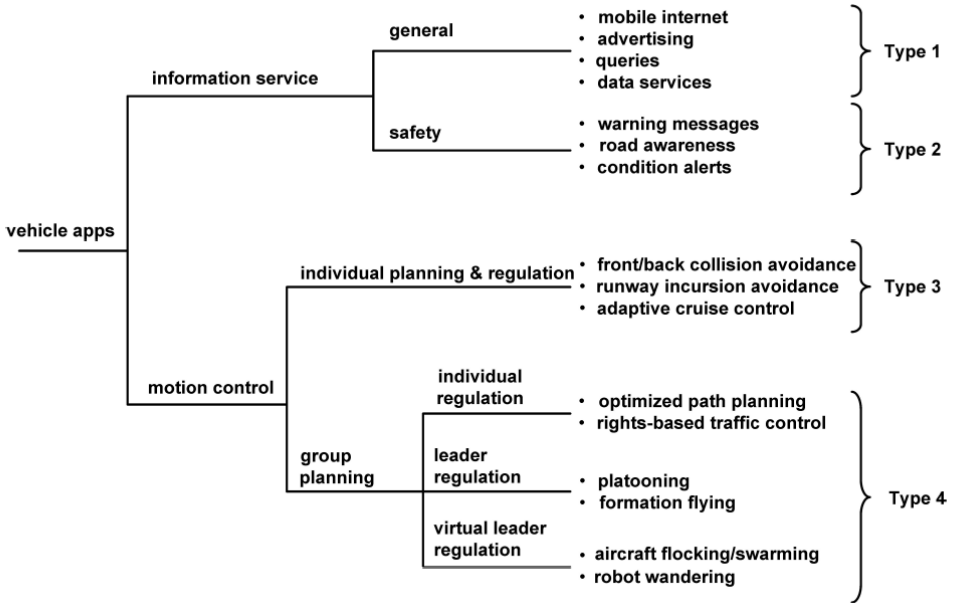


Figure 2.21: Inter-Vehicle communication types [13].

2.4.1.1 VDTN Projects

Vehicular networks can provide network functionalities for applications in several areas: report traffic conditions and accident alerts to a cooperative collision warning, weather reports (ice, fog, wind), information about local free parking, road pavement defects, monitoring vehicles pollution or even advertising and information collection. This kind of networks emerged as an alternative to offer asynchronous cost-effective access to the Internet in remote rural areas, providing non-real time services such as electronic mail, voice mail, web access, telemedicine, environmental monitoring and other applications to collect information. Several projects have been made in VDTNs.

KioskNet [60] was developed at the University of Waterloo and consists of a set of kiosks that use mechanical backhaul as the primary means of communication to the Internet. It is a system to provide very low cost Internet access to rural villages. It is based on the concept of DTN(DTNRG implementation) and uses vehicles, such as buses, to cross data between village kiosks and Internet gateways in nearby urban centers. This project was deployed with just one prototype(car) and one gateway in Anandapuram, a village in South India. It used Wi-Fi(IEEE 802.11a/b/g) for communication between kiosk and prototype. Kiosk controllers would also provide connectivity by other means, such as GPRS, Short Message Service (SMS) and Very-small-aperture terminal (VSAT).

CarTel [61] is a distributed software system that allow to collect and deliver data from mobile sensors. CarTel was tested with 6 vehicles in Boston and used the Epidemic routing protocol. This project was tested with Wi-Fi access points.

Environmental Monitoring in Metropolitan Areas (EMMA) [62][63] is a decentralized architecture for area-wide measurement of air pollutants, a system that allow an continuously acquire environmental data in urban areas. The measured values are exchanged between different vehicles. Since vehicles only meet each other sporadically, techniques from the fields of Car2X (Car2Car and Car2Infrastructure) communications and Delay Tolerant Networks are used for data exchange. This project showed that a public transport network can make part of an efficient and economic solution for environmental monitoring, nevertheless this project used IEEE 802.11b technology to communicate between nodes gateways or relays.

TomTom [64] uses a GPRS bidirectional channel to deliver traffic information and other important information to the receptor installed in the vehicles with a periodicity of 3 minutes. This information allows an accurate estimate of travel times that can be used to select the fastest route.

DieselNet [65] [66] consists of 40 university buses carrying IEEE 802.11b technology during 58 days in Amherst, MA. In this project it was tested RAPID and MaxProp routing protocols. Experimental vehicular deployment allows to take into account issues such as delay, propagation, CPU usage and other possibly unexpected issues that might arise. Both articles propose interesting routing ideas, with RAPID focusing on the intentionality of routing decisions (i.e., making decisions which intentionally maximize the marginal utility according to the objective function) while MaxProp strives to maximize the delivery probability by optimizing the scheduling of packets to be transmitted and droped. Though all these works advance with interesting ideas, we believe it is not enough. Routing protocols such as RAPID and MaxProp, like many others, focus on connecting vehicles in the network and do not tackle the specific problem of connecting vehicles to a wired network.

Table 2.1 presents some VDTN projects performed as well as their applications. As can be observed from the previous description, these projects considered Wi-Fi and cellular technologies, and never tested DTNs on a real platform with vehicular technology, the IEEE 802.11p.

2.4.2 Summary

This chapter presented an overview of vehicular networks based on the literature. The basic concepts were presented, familiarizing the reader with the subject. The first part focused on the architecture, routing protocols and several technical challenges of VANETs.

Then, it was introduced the concept of DTN which is the base concept of this dissertation. DTN architecture and its applications were described as well as several routing protocols challenges were presented.

In the last part it was focused on the concept of DTN introduced in VANETs. The main applications and advantages were presented. Several projects in VDTNs were also referred to show the work that has been made in this specific area.

Project	**Applications**	**Protocol Stack**	**Routing Protocols**
KioskNet [60]	Internet access for rural sites	DTN standard stack	Epidemic
DieselNet [65]	Internet access for rural buses	DTN standard stack	MaxProp, RAPID
VDTN [67]	Internet access for rural vehicles	Bundle layer below network layer. Separate data and protocol planes	Epidemic, Spray-and-wait
CarTel [61]	Detection of road pavement defects	Mule adaptation layer below network layer	Static, epidemic
EMMA [62]	Pollution measurements, traffic information	DTN standard stack	Epidemic
CONDOR [68]	Email, web browsing, IRC, voice mail	DTN standard stack	Static
C2C-CC [69]	Safety, traffic efficiency, infotainment	Standard OSI stack	Geographic routing
TomTom [64]	Navigation, traffic information	GSM	Through infrastructure

Table 2.1: VDTN projects's characteristics [18].

Chapter 3

DTN Implementations

3.1 Introduction

This chapter aims to show some of the real implementations of the Bundle Protocol which were created for DTNs and their main features. Special focus will be given to the implementation of IBR-DTN whose study will be presented along this dissertation.

In the second section of this chapter we will concentrate particularly on the IBR-DTN's basic architecture and its applications.

3.2 Overview Approaches

The Delay-Tolerant Networking Research Group (DTNRG), chartered as a part of the IETF, is working on two protocols whose source code is available in the Internet. The group produced the Bundle Protocol (BP) which is the central issue of this research group. They are engaged in the architectural and protocol design principles to be used in challenging environments where there is no end-to-end path. Licklider Transmission Protocol (LTP) [70] is a point-to-point protocol for very high delay environments.

Several implementations of DTN architecture were created:

- **ION:** Interplanetary Overlay Network (ION) software distribution is an implementation of DTN architecture which is described in Internet RFC 4838. This distribution includes implementations of the BP, the LTP, most of the Bundle Security Protocol (BSP), and two Consultative Committee for Space Data Systems (CCSDS) application protocols that have been adapted to run over the BP/LTP stack.

  According to [71], some features of this implementation are: Compressed bundle header encoding for efficient bandwidth use, bundle streaming service for streaming audio, video, and telemetry over DTN, aggregate custody signaling to control uplink bandwidth consumption, Delay-Tolerant Payload Conditioning plus application data aggregation and elision, and an adaptation of the *DTNperf* performance monitoring tool.

- **Bytewalla:** The main objective of this project is to connect African rural villages by using Android phones with DTN. According to [72], the idea is that people

carry data among cities with their Android mobile phones.

- **DT-Talkie:** It is a voice messaging application that enables mobile users to communicate over infrastructure-less and challenged environments in the walkie-talkie fashion. DT-Talkie [73] was primarily implemented for Maemo based Nokia Internet Tablet and supports both one-to-one and group communication. This application is available to heterogeneous endpoints like Mac, Linux PC, Symbian and Openmoko.

- **POSTELLATION:** It is an implementation [74] that uses the BP, enables Web/ SOA application developers to use DTN "transparently" and also allows a much larger number of end-users to use DTN, developing a community and applications.

- **DTN2:** It is designed as an experimental platform and it provides a robust and flexible software framework for experimentation, extension, and real-world deployment [75][76].

- **JDTN:** It is a Java-based implementation of DTN, as embodied in RFC 5050 - BP, and RFC 5326 - LTP. This implementation was developed for mobile platforms, such as Android.

- **IBR-DTN:** It is a C++ implementation of BP designed for embedded systems. It supports the TCP and User Datagram Protocol (UDP) convergence layers, the BSP and Internet Protocol Neighbor Discovery (IPND) specifications. This implementation as well as DTN2 were studied in a previous project. Several tests [35] showed that the IBR-DTN implementation had a better performance, and so it will be the basis of this project. More details about IBR-DTN will be given in the next section.

To summarize what has just been described, Table 3.1 presents the main implementations referred above as well as their characteristics and applications:

DTN Implementations	**OS**	**Programming language**	**Security Support**	**Applications**	**Routing**
DTN2	Linux, Solaris, Linux on PDA(ARM), MacOS, RAPIS	C++	OpenSSL and partial support for BSP	*dtnping, dtnsend/recv and dtncp/dtncpd*	Table-based, Bonjour, PRoPHET, DTLSR, epidemic, external routing via XML
ION	Linux, MacOS, FreeBSD, Solaris RTEMS e VxWorks	C	(non specified)	Space flights and support for Bundle Streaming Service (BSS)	Contract Graph Routing (CGR)
IBR-DTN	OpenWRT, Debian/Ubuntu, MIPS, Debian ARM platforms, MacOS X, Gentoo Linux, Windows and Android	C++	4 levels (none, Bundle Authentication Block (BAB), Encrypted bundles, Certificate Authority(CA))	*dtnping, dtnsend/recv, dtninbox/dtnoutbox, dtnstream, dtntrigger, dtntracepath*	PRoPHET, Epidemic and Flooding
Bytewalla	Android	Android SDK 1.6	Support for BSP	*ping* and Email	Static routing Bytewalla v.1 and PRoPHET
DT-Talkie	Symbian OS	Symbian OS SDK	(non specified)	Realtime voice communication	Flooding
POSTELLATION	Windows, MacOS, Linux, RTMES and Realtime and Embedded systems	C	TCP-TLS support	*dtnping/dtnpong, dtnsend/recv*, HTTP/HTTPS Proxy, RSS and video streaming	(non specified)

Table 3.1: DTN Implementations

3.3 IBR-DTN

While delay tolerant networking itself is just a concept, a generalized protocol for DTNs, known as the BP (describes the end-to-end protocol, bundle block formats and other specifications of the exchanging of information in DTN), has been specified in RFC5050 by the IETF Delay-Tolerant Networking Research Group (DTNRG)[14]. As IBR-DTN is RFC5050 compliant, it is completely operable with DTN2 and ION.

IBR-DTN is a reference implementation of DTN protocol developed at the Institut für Betriebssysteme und Rechnerverbund (IBR), Germany [76]. It is a module-based architecture with miscellaneous interfaces, and it makes possible to change functionalities like routing or storage of bundle just by inheriting a specific class. This implementation supports the TCP and UDP convergence layers as transport, the Bundle Security protocol and IPND neighbor discovery specifications. Depending on the requirements of the application, the storage module allows bundles to be stored on disk or in a volatile memory. To discover nodes in a local network, IBR-DTN includes the IP neighbor discovery mechanism and a peer-to-peer name resolution service.

When IBR-DTN began to be developed, it was designed for resource-constrained embedded platforms, to run on OpenWRT [77], a Linux distribution built for embedded systems. As a consequence, IBR-DTN is completely compatible with uClibc [78], a C standard library for Linux kernel-based operating systems for embedded systems and mobile devices.

The IBR-DTN software is available for OpenWRT, Debian/Ubuntu, MIPS, Debian ARM platforms, MacOS X, Gentoo Linux, Windows and since recently also Android, allowing the full bundle protocol implementation to be used on Android smartphones.

3.3.1 Architecture Overview

The main goals of IBR-DTN are to use the minimum external requirements such as libraries, to avoid problems with dependencies, and to keep the software implementation small and as modularized as possible. This allows it to keep the overhead small by using just the modules involved and also a better abstraction of the tasks involved like routing, reception, storage and transmission.

IBR-DTN uses an event mechanism shown in figure 3.1 that allows the coupling of the modules by sending events to each other over an event switch.

The following subsections will briefly explain the different available modules of the IBR-DTN architecture.

3.3.1.1 Event Switch

Event Switch is the central module of this architecture. It is responsible for dispatching raised events to/from one module to another, allowing the modules to communicate over events. IBR-DTN is capable of reaching a high degree of concurrency between the several daemon modules due to its capacity of queuing events to a private queue of the module's thread. The standard modules of IBR-DTN are built to allow the creation of applications. Modules can raise and receive events to interact with other parts of the daemon. This implementation has standard events related to storage and routing of bundles, and events regarding the availability of nodes.

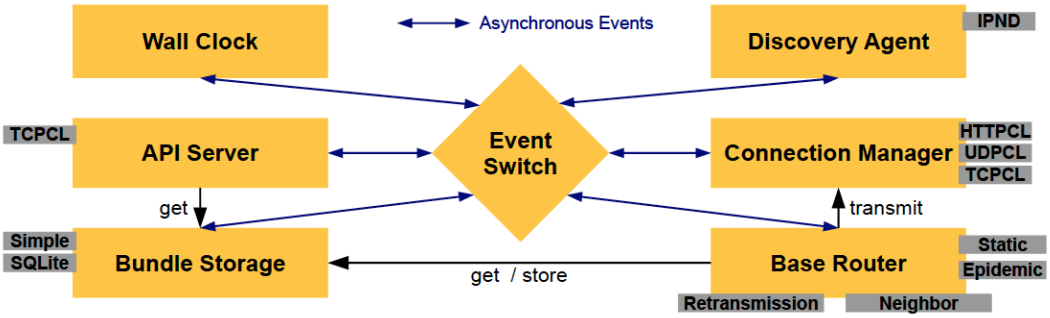


Figure 3.1: IBR-DTN architecture [14]

3.3.1.2 Discovery Agent

The neighbor discovery module generates events whenever new nodes become available, making the routing modules check if there are any bundles to send to the new node. Two different neighbor discovery protocols are implemented: IPND version 1 and 2 [79], and a compatible version with DTN2. This module is responsible for controlling the availability of neighbors.

3.3.1.3 Connection Manager

The Connection Manager handles the lower level protocols IBR-DTN uses to communicate with other DTN nodes. It manages instances of the modules (based on the configuration file), which provide connectivity between the DTN daemons. These modules are the convergence layers, where each layer provides a different interface to transfer the bundles to other nodes. Events are created in case of failed transfers or incoming bundles to be stored in Bundle Storage. IBR-DTN has four implemented convergence layers:

- **TCP Convergence Layer**, compatible with IETF draft, uses handshake mechanism between the daemons. It is equipped with the capability of splitting bundles into segments, which are then acknowledged by the receiving daemon.
- **UDP Convergence Layer** is compatible with IETF draft and the bundles have a size limit to fit in a single UDP datagram.
- **Hypertext Transfer Protocol (HTTP) Convergence Layer** is based on *libcurl* [80] and uses an HTTP server to send and receive bundles.
- **LoWPersonal Area Network (PAN) Convergence Layer** supports the 802.15.4 MAC protocol which is used in small sensor networks.

3.3.1.4 Base Router

This module is responsible for the control of different routing modules interacting with the Bundle Storage and the Discovery Agent modules. It receives events about

nodes connections and the connectivity losses, as well as about bundles arriving to storage. When a routing module wants to send a bundle, it requests the desired convergence layer to Connection Manager to make the transfer. There are several routing modules included in IBR-DTN:

- **Static connections** configured *a priori*. They admit that all configured links are once and for all available.
- **Forward bundles** to the available neighbor nodes discovered by the Discovery agent.
- **Epidemic** routing. IBR-DTN uses Boomfilter mechanism [81] instead of summary vectors. This protocol manages a clean vector capable of deleting delivery bundles from nodes.
- **Flooding** routing scheme is a simple routing algorithm in which every received packet is sent through every outgoing link except the one it arrived in.
- **PRoPHET** routing protocol uses an algorithm which tries to explore the non-randomness of real-world encounters of nodes by maintaining a set of probabilities for successful bundle delivery to known destinations in the DTN (delivery predictabilities). It replicates messages during opportunistic encounters only if the relay node that does not have the message seems to have a better chance of delivering it.

3.3.1.5 IBR-DTN API

It is an application interface based on TCP or Unix Domain Sockets. The frequent re-implementation of bundle streaming protocol in each application can be very difficult; that is why a library is linked to applications simplifying the creation of bundles. A great advantage of this approach is that all supported features of the daemon are immediately ready but, on the other hand, it does not support out-of-band messages. So configuration on-real time is not possible.

3.3.1.6 Bundle Storage

The store-and-forward mechanism makes the existence of a storage necessary to buffer the bundles for possible long intervals of time. The Bundle Storage module was created to provide the needs to the previous mechanism. IBR-DTN has three types of storage that may be used:

- The memory-based storage is a non-persistent storage for access, but in the cost of higher memory usage. The bundles are stored in the Random Access Memory (RAM).
- File Based Storage is a persistent storage, which is used when a path is set and all bundles will be persistently stored in that location based on simple files. Compared to the non-persistent storage referred above, in this case the bundles are saved on disk and their existence survives to daemon's shutdowns and losses

of power. If the memory is not used to store bundles, the daemon requires much less memory to operate.

- SQLite storage uses an SQLite [82] database. This approach is most commonly used in complex routing protocols.

3.3.1.7 Wall Clock

There is an abstraction of a global time inside IBR-DTN based on the host clock which is managed by the Wall Clock. The timestamps in DTN count the seconds since 1/1/2000. This module also sends a global time tick event every second, which is delivered through the Discovery Agent to all other modules.

3.3.2 Security

The security in IBR-DTN is ensured by BSP and it is divided in the next four levels, which can be configured in the configuration file:

- 0 = no constraints (default).
- 1 = accepts only Bundle authentication Block (BAB) authenticated bundles.
- 2 = accepts only encrypted bundles.
- 3 = accepts only BAB authenticated and encrypted bundles.

3.3.3 Applications (Tools)

IBR-DTN has several implemented applications:

- The ***dtnping*** command is used to test the connection between two machines. A node sends out bundles to a specific DTN EID and waits until it receives a bundle with the same payload as answer (indicating the response time). The ***dtntracepath*** allows the visualization of the route between different nodes.
- The commands ***dtnsend and dtnrecv*** can be used together for transferring files between DTN nodes. *dtnsend* creates a bundle and sends it to the destination indicating in the command the source and destination nodes, the file and its path. *dtnrecv* needs to be executed in the receiver node and prints in the screen the bundle when it reaches the destination.
- ***dtnstream*** allows to receive, for instance, an internet radio stream in one DTN node and receives it with another node.
- ***dtntrigger*** is a small utility that can be seen as a lightweight API alternative for scripts.
- ***dtninbox and dtnoutbox*** can be used together to transfer files between two directories on different machines. The files into a specific folder will be sent after executing the command *dtnoutbox* and, after the transmission, they will reach the destination going to *dtninbox* folder.

- ***dtninbox and dtnsend*** can also be used together instead of using *dtnoutbox*.

3.4 Summary

The first part of this chapter aimed to describe DTN implementations giving special attention to the implementation of IBR-DTN.

Concerning the IBR-DTN, the study was carried out taking into consideration the several modules of its architecture as well as their functions. We also focused on IBR-DTN applications and mentioned some security aspects.

Chapter 4

Integration in Experimental Testbed

4.1 Introduction

In the previous chapter we presented DTN implementations, in particular the IBR-DTN, which we used during this work.

This chapter will present the steps taken to enable DTN to work in the real platform in a network of vehicles. It describes the equipment used, logs extracted to study the metrics, technical challenges, problems detected and the interaction with other devices.

4.2 Equipment Used and Parameters

The following subsections present the equipment used in the scenarios described in the chapter 5 to test DTN implementations on VANETs. The Operative System (OS) used in the boards is also introduced.

4.2.1 NetRider board

Figure 4.1 shows an OBU developed in our group, which is an intelligent router that allows the vehicles to communicate through several technologies by using IEEE 802.11g, IEEE 802.11p and GPRS/ 3G/ 4G. It allows vehicle-to-vehicle and vehicle-to-infrastructure communication. This board has the following main specifications:

- Internet connectivity via 3G/4G *backhaul* for up to 10 simultaneous devices.
- Low latency (typ. $<$ 20 ms using IEEE 802.11p).
- High-gain external antennas for each technology.
- Millisecond time synchronization through GPS.
- Low power consumption ($<$ 6 W).
- Automatic standby and wake up depending on the state of the engine.

- Delayed standby upon engine stop.
- Low standby power consumption (< 0.05 W).
- Automatic connection management for cost-effective connectivity.
- Wireless mesh capability for data communications in a vehicular network.
- Seamless mobility between IEEE 802.11p access points (RSUs).

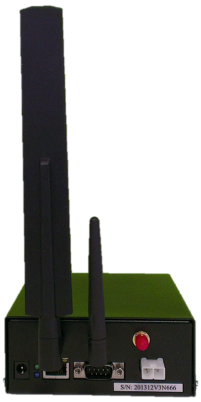


Figure 4.1: Board for vehicular communication [14].

The board is built with the following hardware:

- Processor module AR71xx.
- Wi-Fi module able for IEEE 802.11a/b/g standards.
- Wi-Fi module able for IEEE 802.11p standard.
- Antenna at 2.4GHz, IEEE 802.11a/b/g standards.
- Antenna at 5.9GHz with magnetic support and extension, IEEE 802.11 p standard - used in vehicles (fig. 4.2).
- Antenna for GPRS/3G/4G.
- GPS GlobalTop (MediaTek MT3329).
- USB flash embedded (8GB).

There are connectors in both sides of the board to connect the four antennas (standard g, p, 3G and GPS) and, in the front side, the power connector, Ethernet port and a serial port (RS-232). Depending on the version, boards may have a Universal Serial Bus (USB) port which can be used to connect an USB flash drive.

Figure 4.2: Standard antenna for IEEE 802.11p used for vehicular communications.

4.2.2 SBC board

Figure 4.3 represents a Single Board Computer (SBC) which was used to test the integration of IBR-DTN with different devices. The SBC GW2358-4 is a member of the Gateworks Cambria Single Board Computer family [15], and it meets the requirements for enterprise and residential network applications. The board has the following main features:

- Intel XScale IXP435 667MHz Processor.
- 128Mbytes DDRII SDRAM Memory.
- 32Mbytes Flash System Memory.
- OpenWrt Linux Board Support Package.
- Two 10/100 Base-TX Ethernet Ports.
- Two v2.0 Host USB Ports.
- Optional GPS and RS485 Serial Port.

Figure 4.3: Single Board Computer [15].

4.2.3 Raspberry Pi

Raspberry Pi (RPi) is a low-cost single-board computer developed in the UK by the Raspberry Foundation. It is capable of doing everything that a computer can

do, including browsing the internet, playing High-definition (HD) video or even word-processing. It can be plugged into a computer monitor, a keyboard or a common mouse [83].

RPi is capable of interacting with the outside world and has been used in a wide array of digital projects, from music machines, parent detectors to weather stations and tweeting birdhouses with infra-red cameras.

In this work it was used to receive data from sensors measuring the environmental conditions. In this chapter it will be shown the integration of RPi (with the IBR-DTN) with other devices too.

Figure 4.4 shows the RPi used which uses the RASPBIAN Debian Wheezy OS.

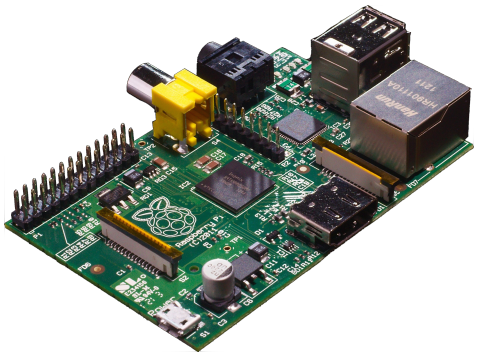


Figure 4.4: Raspberry Pi [16]

4.2.4 Tablet

A tablet was used because it is highly portable and it can easily loose the connection to the Internet. So, DTN can be a great solution to these communication losses. For this work, a Tablet Samsung GalaxyTab will be used to test the software IBR-DTN in the platform.

4.2.5 Parameters

The scenarios tested in the laboratory are performed with the parameters represented in the Table 4.1.

UWME is an application that uses multi-channel for transmission, which may be divided in two types of channels specified in the standard, Control Channel (CCH) and Service Channel (SCH). In this access, the device stays on CCH continuously and a request is sent to access the SCH when it is necessary to be used.

The bit rate was set to 27 Mbps to have guaranteed delivery of all bundles in order to allow the test of different metrics, and also to make sure that some boards were not in the range of others. The TXPower was set to 23 DBm. The number of nodes varied with the scenario, but in the laboratory we used scenarios with a maximum of 3 nodes.

Different parameters were used in Leixões testbed. The bit rate was changed to 6 Mbps in order to have a larger range, and the number of nodes increased to 28 nodes with 3 RSUs and 25 OBUs as presented in Table 4.2.

Parameters	Value
Channel Number	174
Mode	Continuous
TxPower	23 DBm
Bit Rate	27 Mbps
Number of Nodes	2 or 3

Table 4.1: Parameters used for tests in the laboratory

Parameters	Value
Channel Number	174
Mode	Continuous
TxPower	23 DBm
Bit Rate	6 Mbps
Number of Nodes	28 (3 RSUs + 25 OBUs)

Table 4.2: Parameters used for tests in the real testbed.

4.3 Operating System: OpenWRT

As it was mentioned before, IBR-DTN is focused on resource-constrained embedded platforms. So, from the beginning it runs on embedded platforms such as Linux-based OpenWRT systems. OpenWRT [77] is a Linux-based router firmware for embedded devices that uses a command line interface but also offers a web-based Graphical User Interface (GUI) interface.

OpenWRT's development environment and build system, known together as OpenWRT Buildroot, is a set of makefiles and patches that automates the process of building a complete Linux-based OpenWRT system and allows users to easily generate both a cross-compilation toolchain and a root filesystem for embedded system OpenWRT Buildroot [84] [17].

The processor used for embedded devices is usually different from the one found in host computers used for building their OpenWRT system images which requires a cross-compilation toolchain. Toolchain runs on a host system, but generates code for a targeted embedded device and its processor's Instruction Set Architecture (ISA). For example, a host system uses x86 and a target system uses MIPS32; the regular compilation toolchain of the host runs on x86 and generates code for x86 architecture, while the cross-compilation toolchain runs on x86 and generates code for the MIPS32 architecture.

OpenWRT Buildroot automates this process to work on the ISA of most embedded devices and host systems.

Figure 4.5 shows the structure of the OpenWRT Buildroot tree (the fields in the second line are created after compilation).

The *tools* field contains all the building instructions to fetch the image building

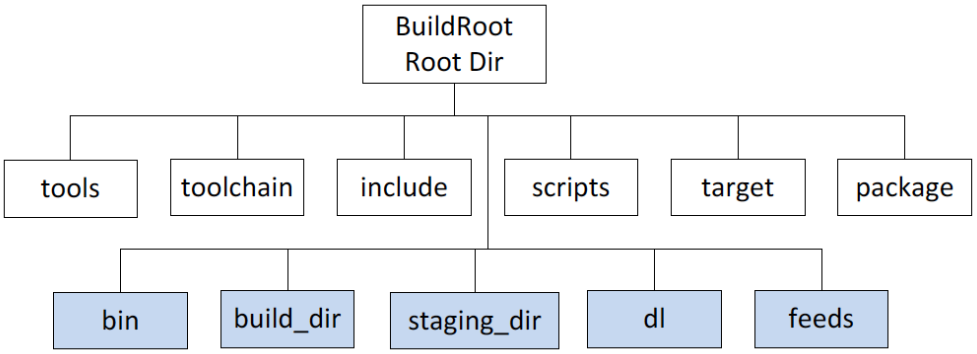


Figure 4.5: OpenWRT source tree [17]

tools. The *toolchain* contains all the building instructions to fetch the kernel headers, the C library, the compiler and the debugger. The *target* contains the building instructions for firmware image generating process and for the kernel building process. The *package* contains the OpenWRT Makefiles and patches for all the main packages. The *bin* is the place where the firmware image will be generated and all the *.ipk* package files will be generated. The *build dir* is the place where all user-space tools will be cross-compiled. The *staging dir* is where the cross-compilation tools will be installed. The *dl* is where the user-space package *tarballs* will be downloaded.

OpenWRT Buildroot provides the following features [84][85]:

- makes it easy to port software across architectures.
- uses kconfig (Linux kernel menuconfig) for the configuration of all options.
- provides an integrated cross-compiler toolchain (gcc, ld, uClibc etc.).
- provides an abstraction for autotools (automake, autoconf), cmake and SCons.
- handles standard OpenWRT image build workflow: downloading, patching, configuration, compilation and packaging.
- provides a number of common fixes for known badly behaving packages.

OpenWRT does not only build system images, but its development environment also offers a mechanism for simplified cross-platform building of OpenWRT software packages. Source code is required for each software package to give a Makefile-like set of building instructions, and an optional set of patches for bug fixes.

In the Netriders boards we used the OpenWRT version Bleeding-Edge, revision 35323.

In the SCB boards we used the OpenWRT version Barrier Breaker.

4.4 Technical Challenges

There were some technical challenges along the project, which the main ones were the following:

- Analysing the code to extract information for logs.
- Mismatching between OpenWRT versions.
- DHT implementation failed to deal with disruptions.
- Synchronization of the boards (clock at the application layer).
- Scripting to perform all the instructions automatically both in laboratory and in Oporto testbed.
- Control and security methods to avoid problems in the testbed.
- Data processing of the results.

4.4.1 IBR-DTN on OBUs

The implementation of IBR-DTN, v.0.10.2 (29-10-2013) (the latest version at that moment), was installed on the OBUs, in OpenWRT OS based on buildroot, to evaluate different scenarios with nodes communicating via standard IEEE 802.11p/g. The boards downloaded the source code from [86], installing the different packages: *ibrcommon*, *ibrdtn*, *ibrdtnd* and *ibrdtn-tools*.

After downloading the software IBR-DTN also to the Virtual Machine (VM) on the PC, it was compiled creating a binary file (set of instructions executed by a processor of a computer). The compilation process required a significant amount of time because this OS was built to operate in devices that usually have low storage and processor capacities. The boards have also these characteristics and the OS has the minor number of libraries to make it lighter as possible; this complicated the installation process which have dependencies on several libraries. Several problems had to be overcome in the compilation of this implementation for the boards, due to missing libraries needed by IBR-DTN and wrong location paths (to find the libraries). After these issues have been solved, the created binary file was transferred to the boards in order to test if everything worked as expected.

When everything worked properly, the code was analyzed to understand where the transfers, receptions and transmissions were performed in order to decide where the added instructions should be included. These instructions allow us to print the logging information to files in order to get information of each event. After some adjustments and compilation of each package, it was time to upload the modified code of the implementation to the boards sending them the binary files created with the compilation.

However, after some time using this version (v.0.10.2) a new version, 0.12.0, was made available by the IBR-DTN research group. Compared to the previous version, this new version has, according to [87], many fundamental improvements in the routing and storage aspects.

There was an upgrade in the *daemon* as follows:

- It was added persistent bundle-set available with SQLite or disk storage, mail convergence-layer, API command for adding or deleting routes and Reverse Path Epidemic for Non-Singleton Bundles in Prophet Routing.

- The acknowledgement set expiration, processing of acknowledgement Set and the decreasing delivery predictability values in PRoPHET routing were fixed.

- PRoPHET was set as a standard protocol in *ibrdtn.conf* (configuration file on IBR-DTN), as well as there was some performance improvements for routing extensions. There were also storage performance problems that needed to be corrected. It was added avoidance re-selecting bundles of aborted transfers due to deletion.

In IBR-DTN *tools*:

- It was added an option to choose destination in *dtntracepath* and Windows support.

- In *dtninbox* and *dtnoutbox* it was removed external dependency to utility 'tar' (*libarchive*).

- In *dtnoutbox* it was added the possibility of reading directly from FAT images by using *libtffs*.

In *ibrdtn library* it was also added the windows support (win32), the persistent bundle-set support, and it was removed bad clock in favor to clock rating and the timezone configuration option. Expiration checking was improved and the hashing of Bundle ID for BloomFilter (now faster) was changed.

In *ibrcommon library* some performance problems were made:

- Windows support (win32).

- Monotonic clock support.

- Link monitoring without netlink (OSX and win32).

- Add support for libnl-2.0.

- Fix compile issues on OS X Maveriks.

This way, we decided to install the new version in the boards in order to prevent some possible errors from occurring. In the beginning we did the same as we had done with previous version: after analyzing the code, the instructions for logging were added to the new code to register information everytime there was some delivery, forward or dispatch. Thus we were able to receive information whenever an event occurred.

After solving some problems with the installation of the software, it was time to start testing the implementation in the laboratory. The results are presented in Chapter 5.

4.4.2 Tools

There were some concerns about *dtninbox/dtnoutbox* while we were using version 0.10.2 because these applications used *tar* archives, and neither the software brought the applications to convert the files nor the boards had that application, a property that was successfully improved in the new version.

The error that used to occur was on sending other type of files. As *dtninbox/ dtnoutbox* use *tar* archives, this explains why it does not work when other files are sent to *dtninbox*: the daemon will deliver them to the *dtninbox* endpoint (it does not know what kind of payload an application expects). According to the daemon, it delivered the bundles. As *dtninbox* expects a *tar* archive, it will probably notice that the payload is not the one expected and so ignores it. The problem was solved in the new version of the implementation which was used for the tests.

4.4.3 Detected Problems

The Distributed Hash Table (DHT) is basically intended to "cross the Internet" [88] without the need of setting up static routes. However, we observed that enabling DHT was preventing the nodes from re-establishing a connection once they came within communication range of each other. It was presented this situation on the mailing list of IBR-DTN and the conclusion was that DHT imposed a large delay in re-establishing connections, which was estimated that it could go up to 30 minutes. This way, the option DHT was disabled.

On the other hand, *blob_path* defines a folder for temporary storage of bundles while they are being created to be sent. If this is not defined, bundles will be processed in memory. In fact, the bundles referred above are stored in this folder, but we observed that they are kept for a too long time, and as a consequence this exceeds the limit of the memory.

4.4.4 Configurations

In the configuration file several modifications were made. In order to have the implementation working successfully in the boards we:

- Changed the number of bundles in transit to 1000 bundles in order to test a large quantity of bundles.

- Enabled *blob_path* for persistent storage of bundles (if not, when the boards restart, all the bundles were lost).

- Enabled the configuration *prefer_direct* to give preference to direct transfer between boards.

- Enabled the *storage_path* that defines a folder for persistent storage of bundles (otherwise the bundles will be discarded when the boards reset).

- The storage limit was set to 20MB in laboratory. In the testbed there were storage limitations, so the storage limit was set to 1MB/3MB in the testbed

As the boards do not synchronize with a clock from the Internet or GPS, the clock was synchronized by setting one of the nodes as master (time reference) and the others as slaves by setting configurations.

Finally, we disabled the DHT option as was explained previously.

4.4.5 Scripting

After the integration of the software in the NetRiders, the implementation was tested in the laboratory with a script that had several functions: to activate the predefined channel, to launch the daemon with the right configuration file, to enable/disable wireless interfaces to make sure that some boards were not in contact when they were not expected to, and, depending on the scenario, to send some files to other boards in the network as will be explained in the next chapter.

The following diagram in figure 4.6 represents the sequence of actions for one of the scenarios tested in the laboratory. It was tested indirect transfer with relay/transport where a bundle was sent from the source node to a relay node while the destination was not available. After 25 seconds waiting, the destination node appears in the range of the relay node, and the message is forwarded reaching the final node. It will be better explained in the next chapter presenting also their results.

After preparing the tests in a laboratory environment, it was time to move to the platform of tests in Leixões Harbor. To carry out these tests there were several changes to be performed in the script. It was needed to pay attention to the storage limit of the bundles. So, a script was made to test automatically the sending of files each 20, 10 and 1 second from each OBU to the server (in the Internet) through RSUs.

4.5 Data Log for Experimental Tests

In order to evaluate the performance of IBR-DTN ,it was necessary to analyze the source code of IBR-DTN to extract information every time a bundle was sent, received or just transmitted through a certain node. This information includes some metrics that will allow to draw some graphs which show several results.

Several metrics were saved in the log files as: source node, destination node, peer node, the time when the event happened, bundle ID, bundle size, size of the folder where the bundles are stored and GPS location where the transfer was processed. The figure 4.7 shows an example of the parameters printed into the log file of a sender node running IBR-DTN at each event. As it can be seen, each line has, in its beginning, a timestamp which was set to have the exact time information about when a specific instruction was executed.

The numbers in the figure 4.7 mean:

1. The first field corresponds to the time the event occurred.

2. The second field can represent 3 actions: R means a local *reception* of the file, i.e, the moment the order to send the files is given; S means *sent* and corresponds to the moment when the transfer was initiated, i.e., after creating the bundle it will be sent from this moment; T means *transferred* and corresponds to the moment the transfer is finished, i.e., the moment when the bundle was completely transferred to another node, its destination or not. In this case the bundle was transferred to *VeniamWorks529*.

3. The third field represents the bundle ID.

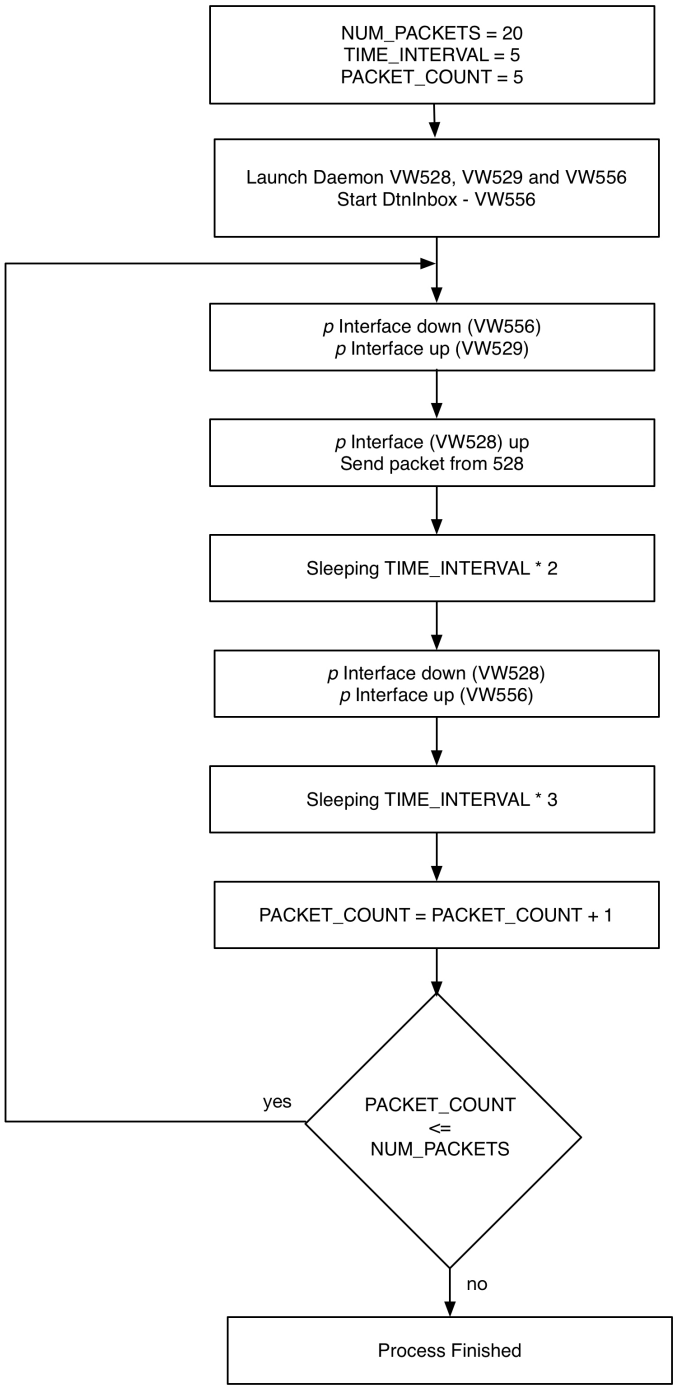


Figure 4.6: Diagram explaining the script used for the fourth laboratory scenario.

4. The fourth field shows the current node which is sending the bundle. In this case it is *VeniamWorks528* board.

5. The fifth field shows the peer node, i.e., the other node (which sends to or receives from this).

6. The sixth field represents the GPS coordinates where the event occurred (in this

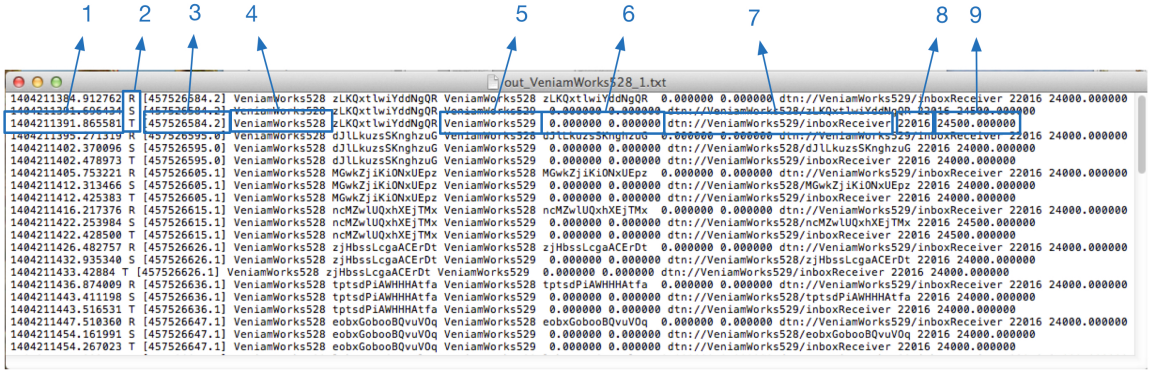


Figure 4.7: Example of Information Present in Data Log

case it is 0.000 0.000 since in the laboratory there was no GPS signal).

7. The seventh field shows the destination node, i.e., the final destination where the present bundle is intended to arrive.

8. The eighth field shows the bundle size (in bytes).

9. The ninth field shows the size of the folder where the bundles are stored.

To have this data set, it was necessary to explore, modify and add instructions to IBR-DTN source code. It was necessary to understand the code in order to find the exact moment when it was given the order to send the bundles, the moment when the bundles begun the transfer to other nodes, and also the moment when the transfer finished.

After having the results from laboratory it was necessary to reduce the amount of logs that would be created in the testbed, reducing the quantity of words and irrelevant information on them in order to avoid excess the limit of memory capacity of the boards.

The logs created were used to generate the final results presented in the next chapter.

4.5.1 Evaluated Metrics

Several values were saved in the log files in order to evaluate the following metrics:

- **Ratio of data successfully delivered**: data delivered to RSUs / data generated by the vehicles.
- **Overhead added to the network**: sum of all data transmitted by a vehicle to other vehicles or RSUs.
- **Delay**: the time it takes from the generation of the packet until the moment it is delivered to an RSU or OBU.
- **Number of Hops:** number of hops each bundle successfully delivered travels in the first path to deliver it to the destination.
- **Useful traffic**: data delivered vs. additional overhead.

4.6 Interaction with other Devices

4.6.1 Heterogeneous Testbed

After testing the IBR-DTN implementation using just the OBUs, and since the results evidenced a high performance, it was necessary to try another scenario, this time with other types of devices integrated in the same network (all of them with the software IBR-DTN): Servers, OBUs, SBCs, Tablet, Raspberry Pi and a Macbook. These devices had different OSs, like OpenWRT, Ubuntu, Debian, MacOS and Android as it is presented in the figure 4.8 between brackets close to each figure of each device.

First, it was compiled in the respective VM (a different version from the one used to compile to NetRiders) the modified code to SBCs, and then installed in all SBCs shown in the figure 4.8.

Afterwards the network was built according to the figure by having two servers: *CloudServer2*, at home, that was accessible through Internet, and *CloudServer1* that was in Instituto de Telecomunicaçoes (IT) network, which had access to the Internet too. Two SBCs called *openwrt3* and *openwrt2*, were established as gateways of IT network, *gw1* and *gw2*, respectively. It is relevant to refer that these boards were not in the range from each other, i.e., they could not communicate directly. The communication between them could just be made by using a relay node, *openwrt1*, which will allow the transfer of information between the two nodes, and works as a Hotspot, creating a network in which are connected the Tablet and Raspberry Pi. The three nodes (*openwrt1, openwrt2 and openwrt3*) create an Ad-hoc network which also included other devices such as an OBU, namely the *VeniamWorks288*, and one Macbook.

Considering the features of OBUs, these boards are able to connect to other devices through several wireless interfaces. Thus, *VeniamWorks288* was connected to the Ad-hoc network through the IEEE 802.11g interface and also to a IEEE 802.11p network which included two other OBUs, *VeniamWorks528* and *VeniamWorks529*. It is important to say that the board *VeniamWorks528* worked as relay, i.e, it created the link between *VeniamWorks288* and *VeniamWorks529* because these were not in the range of each other.

Several tests will be done in this network to attest the real integration of all present devices with the software IBR-DTN.

4.6.2 Time Synchronization

Once created all the testbed, the synchronization was difficult since there was no GPS signal inside the laboratory. This problem did not happen in Oporto testbed. Another reason why the process was difficult was the fact that there were different types of boards, but not all of them had a common behavior. First, it was tried to synchronize the boards through the application layer, with the clock of IBR-DTN. This worked properly with the OBUs. Nevertheless, the same did not happen with SBCs and, even configured as slaves, they could not synchronize through the master that was one of the OBUs.

Several synchronization mechanisms were considered, besides the synchronization through the application layer and through GPS, in order to have the same time in all

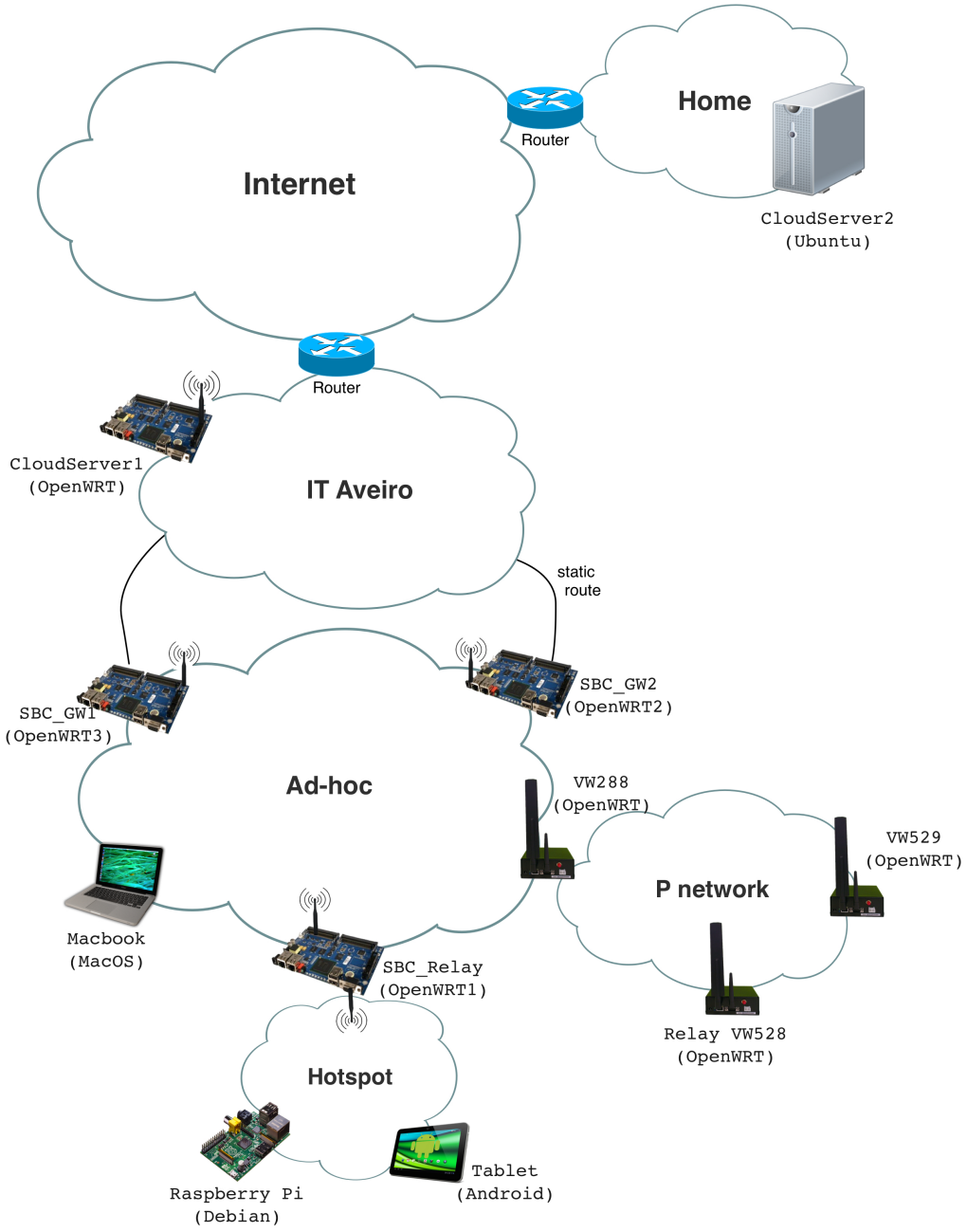


Figure 4.8: Network scheme of the heterogeneous testbed.

devices present in the network, such as Precision Time Protocol daemon (PTPD) [89] and Network Time Protocol (NTP) [90].

In the end, it was choosen NTP to synchronize the boards because it is a native protocol in OpenWRT installation, while PTPD may not work in some versions of this OS.

NTP [91] [92] is a networking protocol to synchronize the clocks between computer systems over packet-switched, variable-latency data networks. This protocol is intended to synchronize all participating devices with UTC with a millisecond precision. It uses an algorithm to select accurate time servers, and is designed to attenuate the

effects of variable network latency. NTP can usually maintain time synchronization within dozens of milliseconds over the public Internet. Asymmetric routes and network congestion can cause errors of 100 ms or more.

NTP is usually described in terms of a client-server protocol, but can easily be used in peer-to-peer relationships where both peers consider the other to be a potential time source. Implementations send and receive timestamps using UDP.

It was necessary to create static routes so that the servers (slaves) might be synchronizey by the master previously defined as *VeniamWorks288* from the network shown in the figure 4.8.

4.7 Integration in Oporto testbed

When everything worked properly in the laboratory it was time to move to a real world platform (Figure 4.9), a testbed with 28 nodes which includes trucks, boats, common vehicles and 3 RSUs.

It was tested the sending of files from each mobile node to a specific server in the Internet. From the performed tests, it were registered the logs of the transmissions in order to evaluate the metrics referred in 4.5.1, which results are in the Chapter 5.

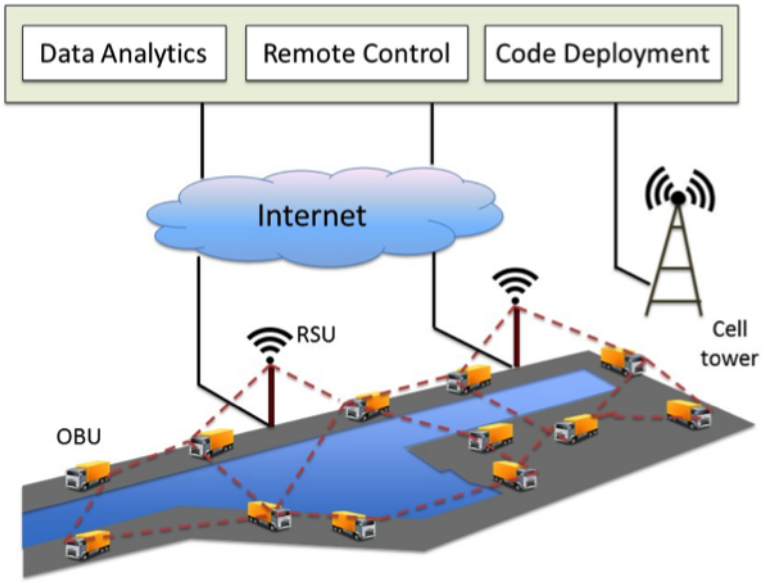


Figure 4.9: Harbor scenario

4.7.1 Control and Security Guarantee

It was necessary to establish a storage limit for the bundle folder. As the boards have a small storage capacity, it was defined a capacity of 1MB of storage in the beginning and, after some performed tests, it was extended do 3MB. This way, when the bundle's folder reached the limit of storage, the bundles were lost which caused

some undesirable results. This fact caused some restrictions which will be explained in the next chapter.

There was another aspect related with the storage which needed to be revised: the quantity of logs printed in each event were too much for the storage capacity of the boards. Thus, it was necessary to compress the logs removing information which was not so important for the results.

In order to prevent the excess of storage limit in the boards, there was another condition: when the global storage capacity of the boards was lower than 25MB, the logs were not written in order to not use the small amount of remaining storage. Later this condition was set to 15MB.

4.7.2 Data Analysis

To analyze the results of the tests performed on the platform as the example shown in Figure 4.7, a script in Matlab was made to read all the data received from the testbed and plot several graphs for the metrics referred above. The graphs will be presented in the Chapter 5.

4.8 Summary

Along this chapter, it was presented the taken steps before start testing the different scenarios to get results about IBR-DTN implementation. It was also presented the difficulties and challenges met working with IBR-DTN software.

The next Chapter will present the results taken from the tests with IBR-DTN both in laboratory and in the real world platform.

Chapter 5

Tests and Results

5.1 Introduction

After having IBR-DTN integrated on the boards, with the challenges overtaken, it was time to evaluate the implementation on a set of different scenarios established. We incremented the number of nodes changing the conditions on each scenario in order to check if everything was running properly.

This chapter presents the scenarios tested, supported by pictures and diagrams which represent the tests. It also shows the experimental results taken from each scenario.

Section 5.2 presents the metrics evaluated in the tests that will be studied in the results.

Section 5.3 presents the laboratory scenarios, as well as their results, while section 5.4 presents the real testbed scenarios and the corresponding results in Leixões' harbor.

5.2 Laboratory Tests

In the laboratory several tests were carried out with the boards without movement. We started with the tests to evaluate IBR-DTN performance.

The first set of tests was performed just with OBUs, being this a first approach test to the real platform of tests. We varied the conditions of the tests namely the bundle sizes (20kB, 200kB and 2MB), the number of nodes and the delay between transfers by enabling/disabling interfaces in each board involved in the transmission.

The second set studied the integration of different devices in the same network. These devices were servers, OBUs, SBCs, a tablet, a Macbook and a Raspberry Pi, all of them having installed and working with IBR-DTN software.

5.2.1 Communication Tests

5.2.1.1 Scenario 1

The first scenario was the simplest scenario tested in the laboratory with two boards. Two static boards were connected via IEEE 802.11p standard as shown in Figure 5.1.

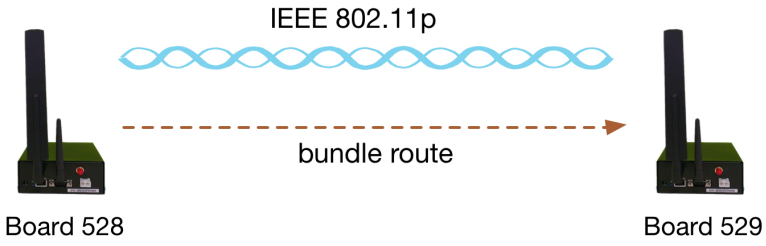


Figure 5.1: Direct transfer

In this scenario the boards, always connected via IEEE 802.11p standard, are in the range of each other as represented in the figure. The main aim of this test is to know if the transmission is correctly processed and, at the same time, to understand through the extracted logs the influence of the bundle size on the transmission delay.

A script was made to automatically run all the experiment. This process is represented in a timeline (Fig. 5.2). The timeline represents only one step of the experience, but it was repeated 20 times for each file size (20kBytes, 200kBytes and 2MB).

This script created a file identified with a sequence number in the emitter node (*VeniamWorks528*). This file has a fixed size to be sent with *dtnsend* tool of IBR-DTN to a specified node (in this case to *VeniamWorks529*). Data logs are registered every time the order to be sent was given, creating an event that is represented by R in Figure 5.2. The letter S represents the moment when the bundle is beginning to be sent to the next node. The time between R and S is called the *pre-send phase* that corresponds to a set of processes that occur in the board, such as the bundle creation, the discovery of new neighbors, the bundle storage in the bundle folder, and its "push" from this folder to be sent to the next node, etc. The letter T represents the moment when the bundle transmission is finished, and this event is also recorded in the logs which allows us to know when the bundle reached the next node. That node may or not be the final destination node. If it is the final node, the *dtninbox* tool extracts the bundle to a specified folder named *inboxFolder*. This way we get the file in the receiver node.

After analyzing the log files obtained for this test, we can represent the results in two graphs. The first graph (fig. 5.3) shows the delay versus file size.

The graph shown in the Figure 5.3 represents the average time the files take to reach the final destination according to the different size. The time represented includes the several phases, i.e., from R to S phase (*pre-send phase*) and from S to T phase, which corresponds to the wireless transmission between the two nodes.

From these results it is possible to conclude that, the bigger the file is, the more time it will take to reach the next node.

The second graph, Figure 5.4, shows the delay versus phase of transmission. In this graph we focus on some details of each transmission, analyzing each phase with different sizes of files. This way it is possible to see that in R - S phase, the time it takes is much higher with the 2MBytes files size than with smaller files. This is completely different in S - T phase, where the time it takes for the different sizes of files is very similar. This can be explained because it was defined a channel with a very

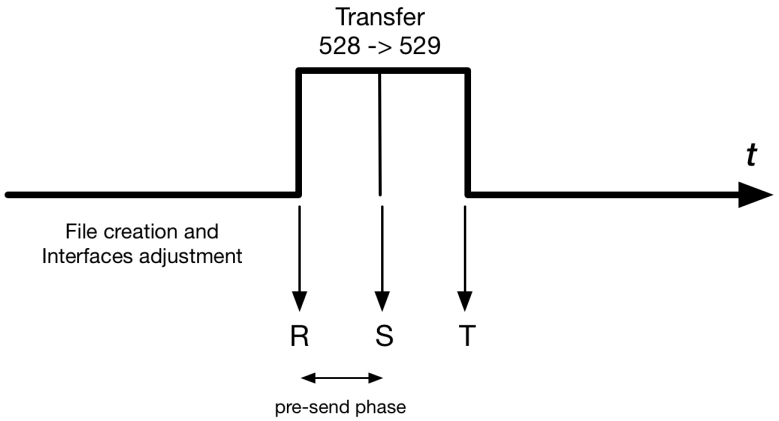


Figure 5.2: Timeline representing each phase of the test

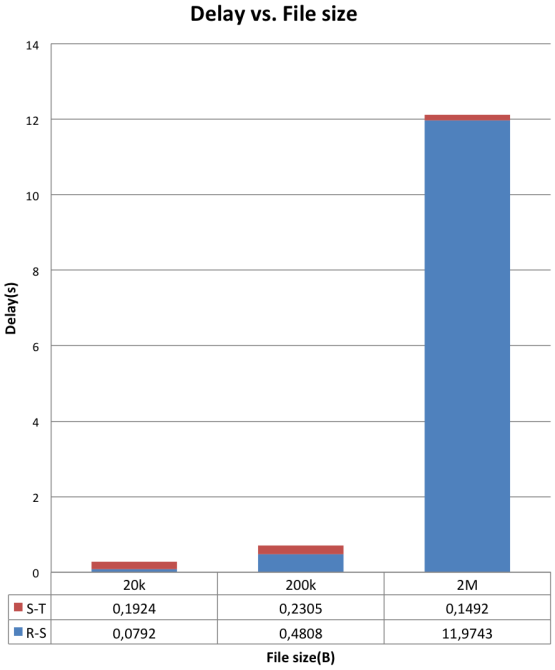


Figure 5.3: Delay versus File Size

high bitrate of 27 Mbps.

With these results we can conclude that, in all the process of transmission, the phase that takes more time is the R - S phase (bundle creation, neighbors discovery, storage in the bundle's folder, etc). On the other hand, the S - T phase (wireless transmission) is much faster and stable (for these file sizes and channel bandwidth). As shown in the Figure 5.4, the obtained confidence intervals were low, which means that the values obtained for 20 times the test was repeated were regular, not having big variations from the average value.

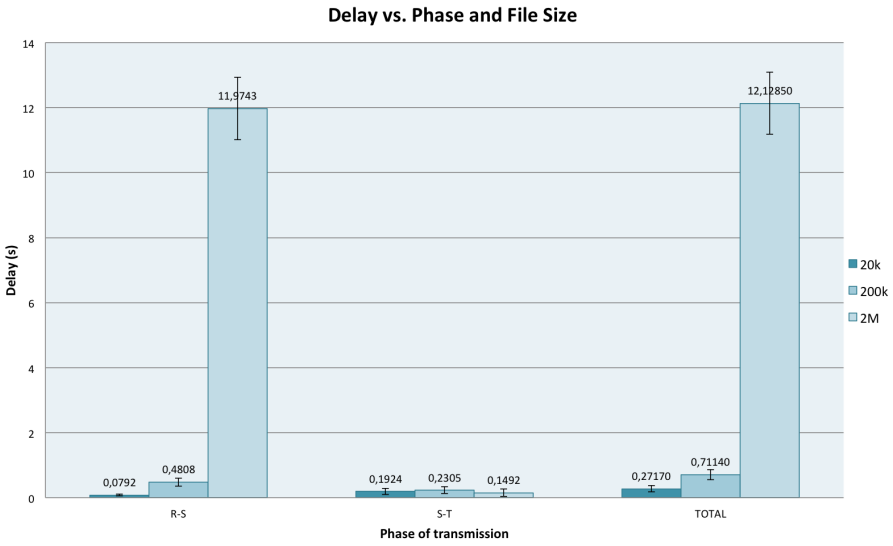


Figure 5.4: Delay versus Phase of Transmission and File Size.

5.2.1.2 Scenario 2

It was also executed a second test with two boards but, in this case, with a disconnection time of 5 seconds. The main aim of this test is to evaluate the real concept of DTN, i.e., the store, carry and forward mechanisms which are the main characteristics of this type of networks. In this scenario the boards were not always connected via IEEE 802.11p standard as represented in Figure 5.5.

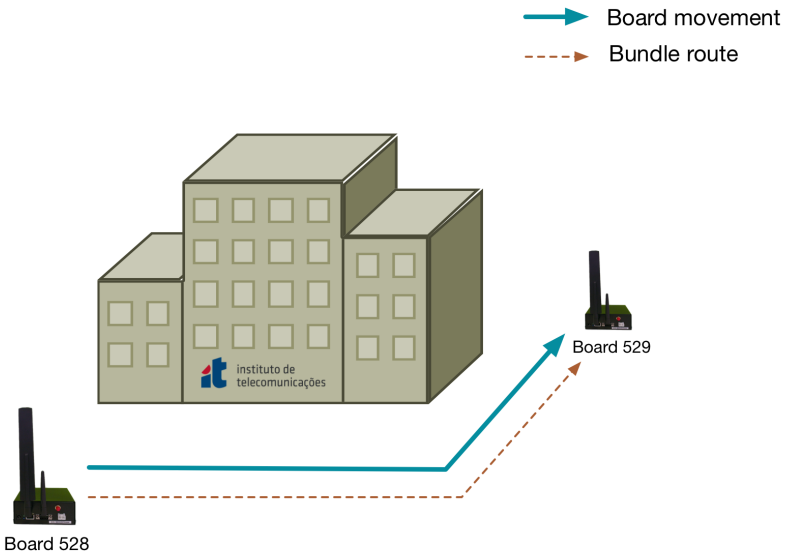


Figure 5.5: Direct Transfer with delay.

According to the image represented in the figure 5.6, we can see that the timeline is similar to the previous one, but it is different when the destination node is not available in the moment the file is sent. After sending the file, the script waits 5 seconds so that

the current node (*VeniamWorks528*) may store the bundle until the final node appears. It should be noted that the represented sequence was repeated 20 times for two different sizes: 20kB and 200kB.

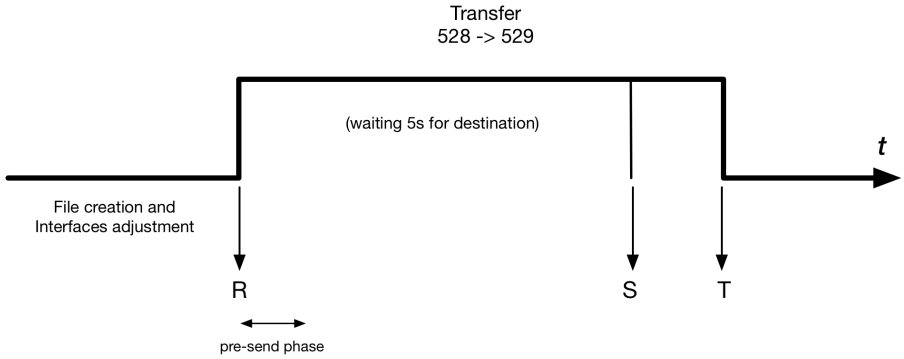


Figure 5.6: Transmission Timeline

To summarize, the event *R* follows the same description as in the first scenario, but the *S* and *T* event are expected to happen 5 seconds later. In this scenario *dtnsend* and *dtninbox* tools are also used in the source and in the reception nodes respectively.

After analyzing the log files obtained for this test, we can represent the results in two graphs. The first graph (Fig. 5.7) shows the delay versus file size.

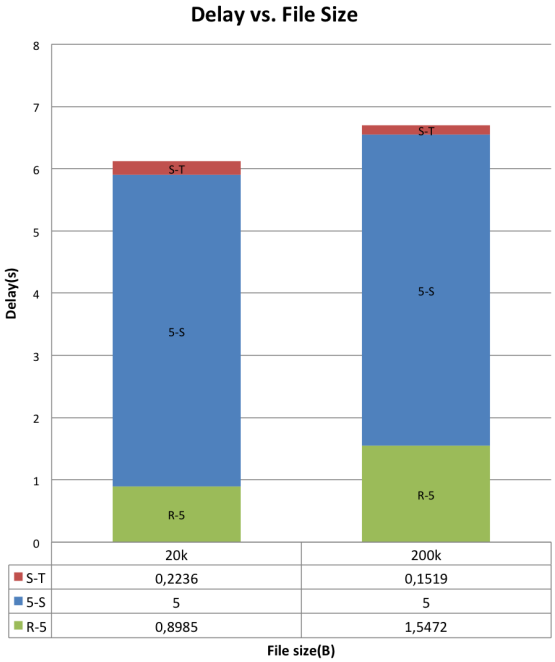


Figure 5.7: Delay versus File Size

The graph shown in the Figure 5.7 represents the time each file takes to reach the final destination according to the different sizes of the files tested. The time represented includes the several phases, i.e., from *R* to *S* phase (which includes *pre-send phase*

and the 5 seconds delay) and from S to T phase which corresponds to the wireless transmission between the two nodes.

As it happened in the previous scenario, from these results, it is possible to conclude that, the bigger the file is, the more time it will take to reach the next node.

The second graph, Figure 5.8, shows the delay versus the phase of transmission. The graph shows similar results as in the figure 5.4 from the first scenario, but there is a new element which is a 5 seconds' delay. This extra delay was added to observe the store, carry and forward mechanism, typical of DTNs.

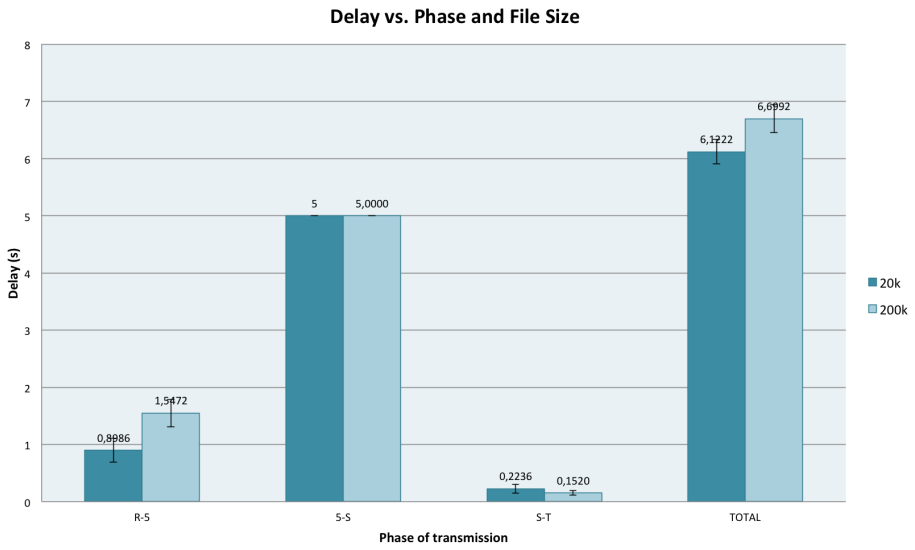


Figure 5.8: Delay versus Phase of Transmission.

This way we can draw the same conclusion as in the previous scenario. Besides, we observe that the addition of a 5 seconds' delay causes no significant differences in the time of the transmission process.

5.2.2 Indirect Transfer

The third scenario is built with three nodes as presented in Figure 5.9. In this scenario the boards *VeniamWorks528* and *VeniamWorks556* are not in the range of each other, but there is another board, *VeniamWorks529*, that works as a relay node and has always a connection via IEEE 802.11p with both boards in the extremes. So, the relay can make the bridge between the two nodes creating a multi-hop connection between the extreme boards.

The main aim of this scenario is to test how IBR-DTN behaved with more than two nodes. The middle node needed to forward the bundles sent by the source to the destination.

The diagram shown in Figure 5.10 represents the function of the script created to make everything automatic. The script was written to send files with a sequence number only from *VeniamWorks528* to *VeniamWorks556* as previously defined. In the diagram it is possible to see the phase related to the first transfer (from *VeniamWorks528*

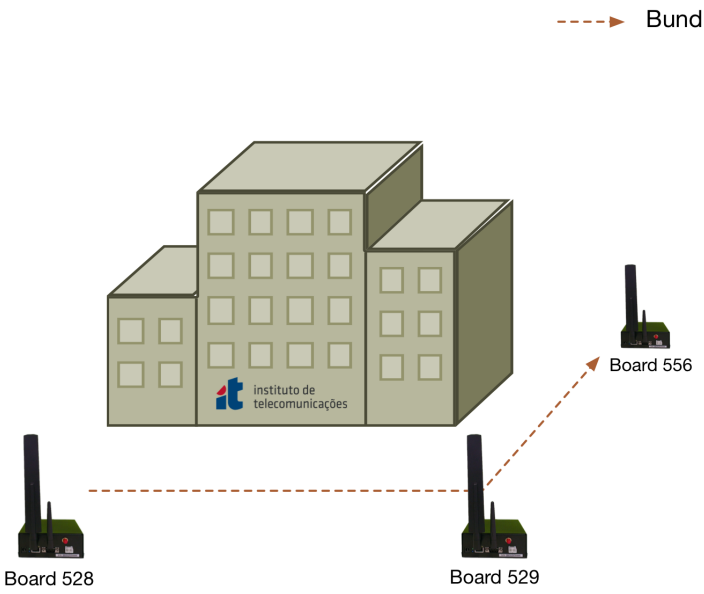


Figure 5.9: Indirect Transference.

to *VeniamWorks529*) and also the second transfer (from *VeniamWorks529* to *VeniamWorks556*). Concerning the first step we can see three events (registered in the data logs): *R1*, *S1* and *T1* which were explained above in the first laboratory scenario. It will be important to emphasize that events *T1* and *R2* coincide as they occur at the same time. For the second transfer it is possible to see that the events *R2*, *S2* and *T2* occur the same way as in the first transfer. This means that, in the relay node, although this node already knows the destination, it takes time running the *pre-send phase* in which a set of processes occur in the board: the bundle creation, the neighbors discovery, the bundle storage in the bundle's folder and its "push" from this folder to be sent to the next node, etc.

The sending is performed by the application *dtnsend* and the reception, in the final node, by *dtninbox* application, both available in IBR-DTN. The former is responsible for the sending, and the latter *dtninbox* is responsible for the extraction of the bundle to a specified folder named *inboxFolder*.

It is important to mention that the actions represented in the timeline were repeated 20 times for each of the three sizes studied: 20kBytes, 200kBytes, 2MB. The results of each of these tests are presented in the following graphs.

The graph shown in Figure 5.11 represents the time each file takes to reach the final destination according to the different sizes of the files. The time represented includes several phases, i.e., *R1* - *S1* phase (*pre-send phase*) and from *S1* - *T1* phase, which corresponds to the wireless transmission between the two nodes of the first transfer from *VeniamWorks528* to *VeniamWorks529*. The graph also shows the second transfer, from *VeniamWorks529* to *VeniamWorks556* which includes the same phases *R2* - *S2* and *S2* - *T2*.

From these results, similarly to the previous scenarios, we see that the bigger the file is, the more time it will take to reach the next node.

The second graph, Figure 5.12, shows the delay versus phase. In this graph it

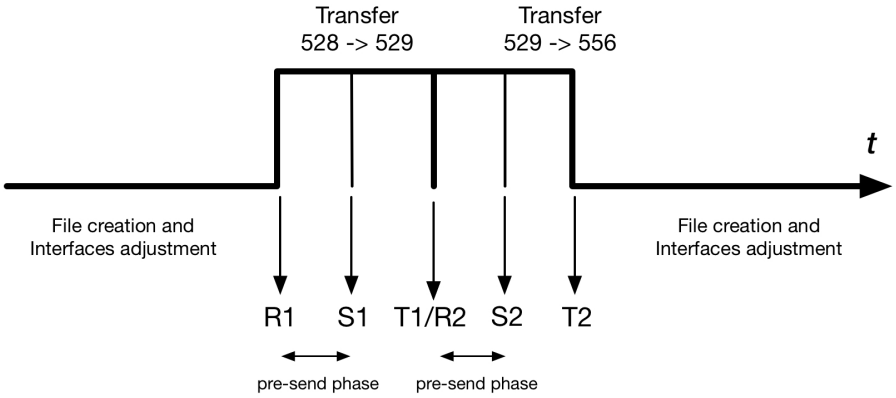


Figure 5.10: Transmission Timeline

.

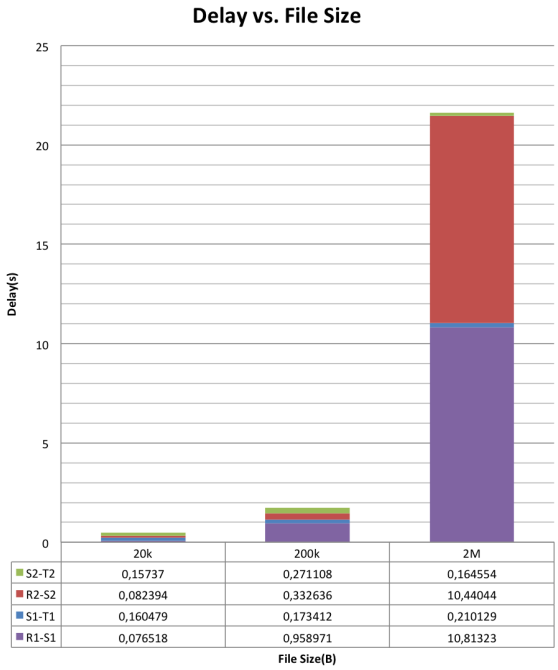


Figure 5.11: Delay versus File Size.

is analyzed each phase for different sizes of files. It is evident that the repetition of phase *R2 - S2* (with the bundle creation, the storage in the bundle's folder and its respective "push" to send) should not happen because the bundle should be immediately forwarded once the final destination was already available. On the other hand, the wireless transmission phases (*S1 - T1* and *S2 - T2*) seemed to happen in a short period of time as expected.

We can state that the repetition of the *pre-send phase*, *R2 - S2*, in the second transfer can be considered a waste of time.

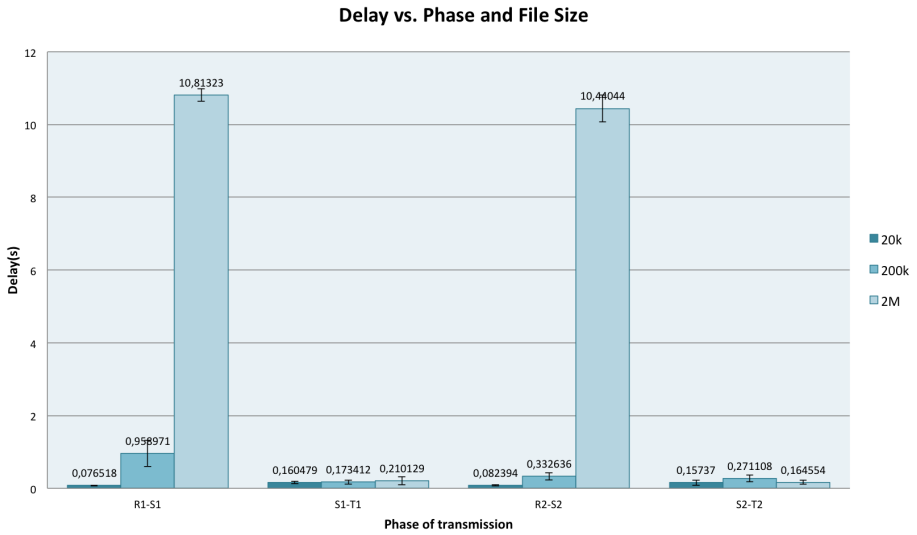


Figure 5.12: Delay versus Phase of Transmission.

5.2.3 Indirect Transfer with relay / Transport

Figure 5.13 illustrates the fourth scenario tested in the laboratory. There are three boards as in the previous scenario, but this time the position of the relay node is different, i.e., in the beginning this node has contact only with the source node and not with the destination. This implies that the relay node needs to store the bundles until the next destination is available, and so it is possible to evaluate the store, carry and forward mechanism which is the main aim of this test.

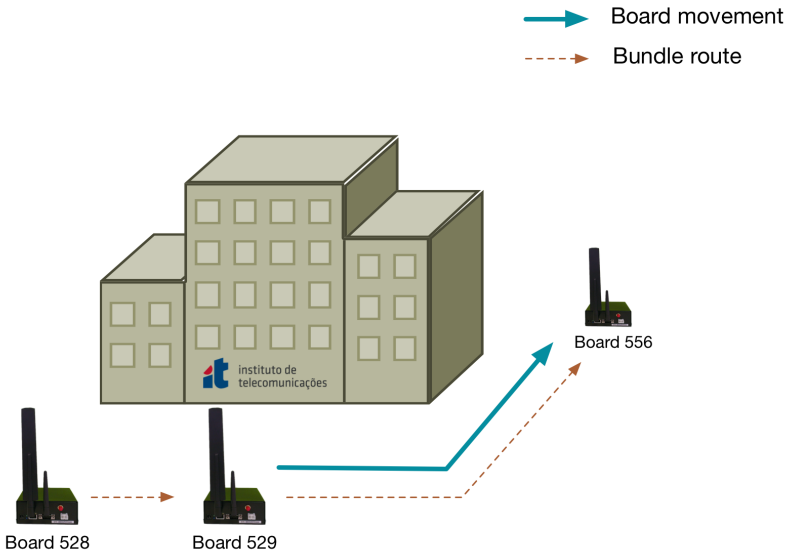


Figure 5.13: Indirect transfer with relay.

The following timeline (Fig. 5.14) whose data was automatically produced by the script, represents what has just been described. We can observe that the source board

(*VeniamWorks528*) creates a bundle which is sent with *dtnsend* tool to the final destination (*VeniamWorks556*). The source node receives the file (at the *R1* event) and, in the *pre-send phase*, creates the bundle, makes the neighbor discovery, stores the bundle in the bundle's folder and makes the "push" of that folder in order to send the bundle to the next node. This moment corresponds to the *S1* event when the bundle begins to be sent. As soon as the bundle reaches the relay node, *VeniamWorks529* (*T1* event), the final destination will not be available yet, so this node needs to "store and carry" this bundle until the final destination becomes available. As this is a controlled scenario, the time it will take until the final destination is within range of the relay node will be 25 seconds. After this time, the final destination will be available and the transmission of the bundle will happen. The *T2* event corresponds to the moment when the bundle reaches the final destination. This reception is possible because of the existence of a *dtninbox* application running in the final node.

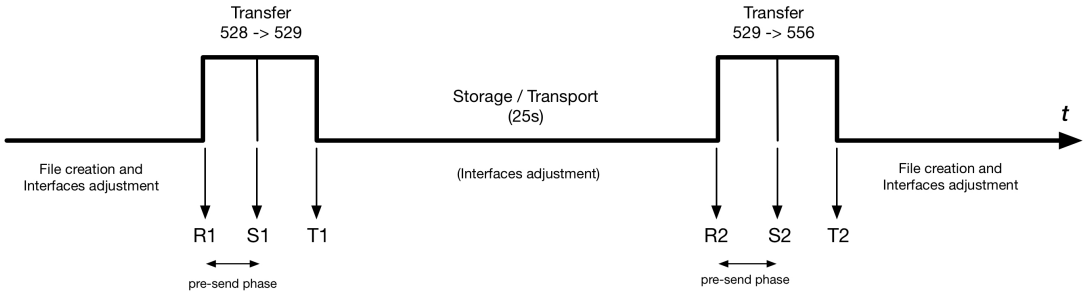


Figure 5.14: Transmission Timeline

The procedures represented in the diagram were automatically repeated 15 times for two different sizes of files: 20kB and 200KB.

After analyzing the log files obtained for this test, we can represent the results in two graphs. The first graph (Fig. 5.15) shows the delay versus file size.

The graph represents the time each file takes to reach the final destination, taking the different sizes of the files into consideration. Although a 25 seconds' delay is added to test the store, carry and forward mechanism, the results show no significant differences in comparison with the previous scenario.

The second graph (fig. 5.16) does not show the 25 seconds' delay above referred to, so that we can see better the times of each transmission phase. The results are as expected, since the R to S phases (of the two transfers) are the ones that take more time in the transmission, and the 25 seconds' delay play no important role in the transmission process.

5.2.4 Integration Results

As it was mentioned before, the second part of the laboratory tests are carried out to test the implementation of IBR-DTN in different devices with different OS and different wireless technologies, as it is represented in Figure 5.17.

Since not all devices have the same applications/tools available and only the *dtnping* tool was available in all of them, this tool was the selected one to be used.

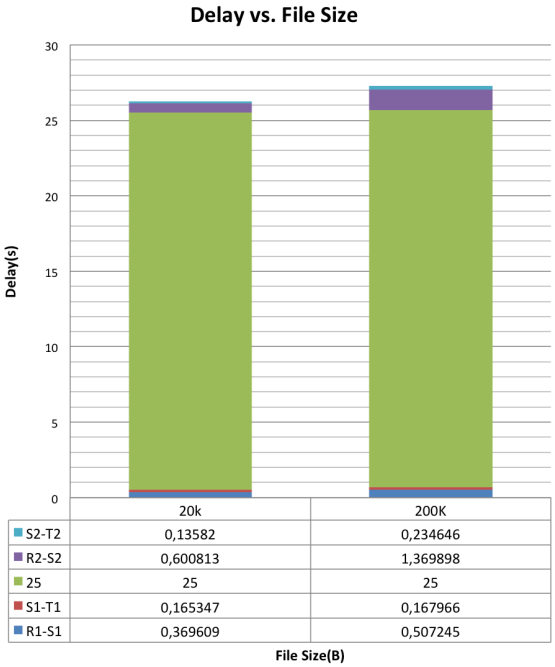


Figure 5.15: Delay versus File Size

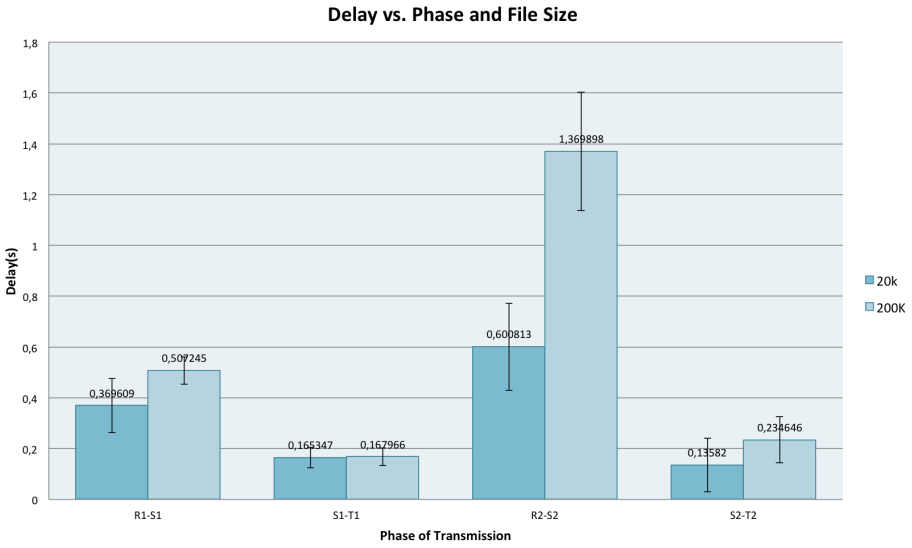


Figure 5.16: Delay versus Phase of Transmission

Several files are sent with *dtnping* in order to know how much time it takes to send a small file (64bytes) from one node to another as it is shown in Figure 5.17. The nodes are randomly chosen. For instance, files are sent from the SBCs to the servers, crossing some of the networks (via IEEE 802.11p and g (Wi-Fi) standards) present between these two devices. All pings are successfully made getting the times for each one.

It should be noted that the test is carried out with all devices shown in Figure 5.17 and everything worked properly.

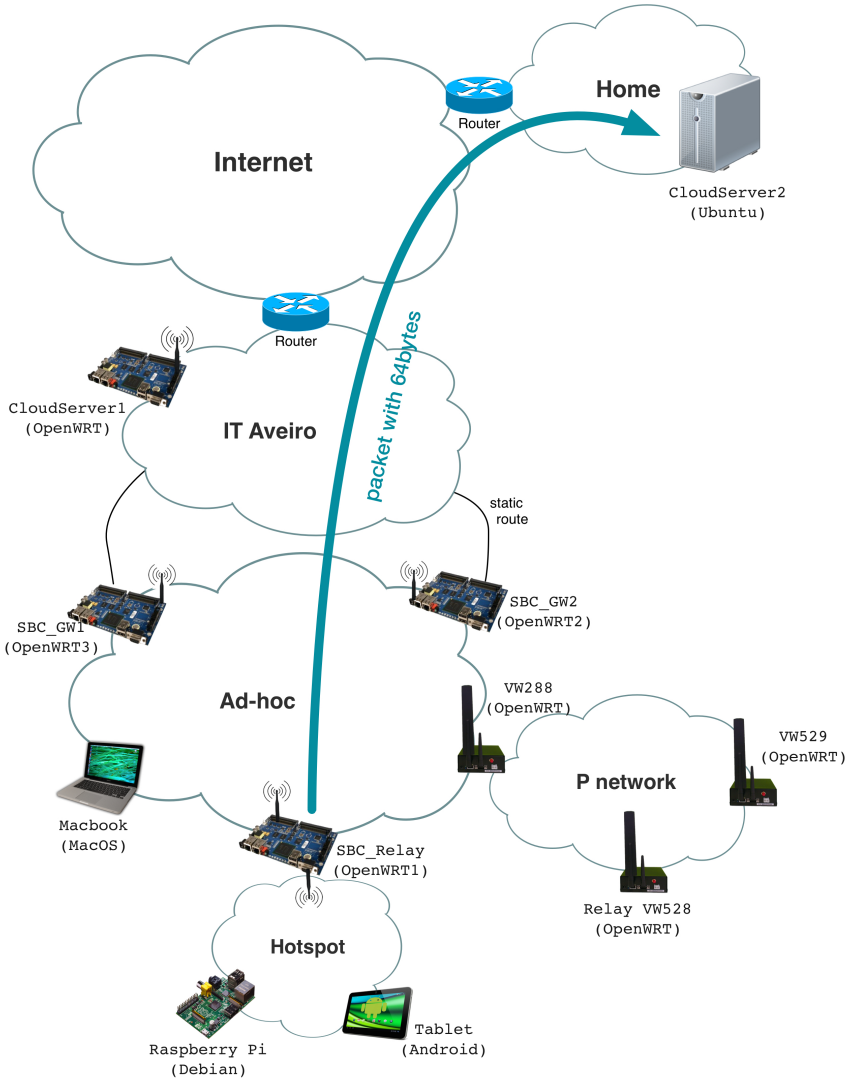


Figure 5.17: Integration in a heterogeneous testbed

The following two graphs show the delay depending on the type of transmission between two different devices having different OSs and wireless technologies, but with the same IBR-DTN implementation.

The first graph, Figure 5.18, illustrates the transmission between boards and servers crossing all networks: Internet, IT network, Ad-hoc and/or P networks depending on the case. This graph shows the results from the different transmissions: from OpenWrt3 to CloudServer1 (*op3-cd1*), from OpenWrt3 to CloudServer2 (*op3-cd2*), from OpenWrt1 to CloudServer1 (*op1-cd1*), from OpenWrt1 to CloudServer2 (*op1-cd2*), from VeniamWorks288 to CloudServer1 (*VW288-cd1*), from VeniamWorks288 to CloudServer2 (*VW288-cd2*), from VeniamWorks528 to CloudServer1 (*VW528-cd1*) and from VeniamWorks528 to CloudServer2 (*VW528-cd2*).

From this graph, it is possible to observe that most of the results are above 300 milliseconds. The values that differ more from the average can be explained with some interference (from the walls in laboratory as well as due to people passing between

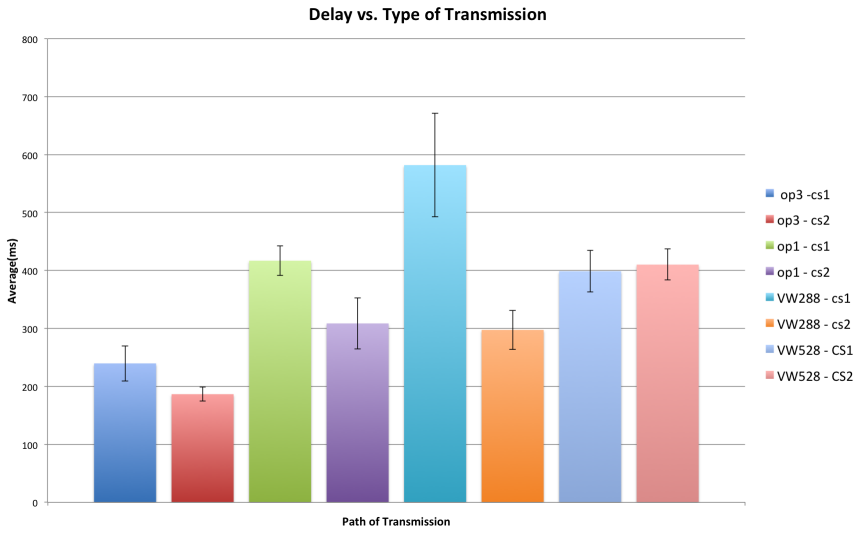


Figure 5.18: Delay versus Path of transmission

the boards during the tests), the congestion of the Internet or IT network, and even due to the heterogeneity of technologies present in this network which needs to cross multiplatform systems.

The second graph, Figure 5.19, shows the exchange of files only between boards this time crossing only Ad-hoc and P networks. This graph shows the results from the different transmissions: from Tablet to VeniamWorks288 (*tablet-VW288*), from Tablet to VeniamWorks528 (*tablet-VW528*), from Tablet to OpenWrt1 (*tablet-op1*), from OpenWrt3 to OpenWrt1 (*op3-op1*), from OpenWrt3 to VeniamWorks288 (*op3-VW288*), from OpenWrt3 to VeniamWorks528 (*op3-VW528*), from OpenWrt1 to OpenWrt1 (*op1-op2*), from OpenWrt1 to OpenWrt3 (*op1-op3*), from VeniamWorks288 to VeniamWorks528 (*VW288-VW528*) and from VeniamWorks528 to op2 (*VW528-op2*).

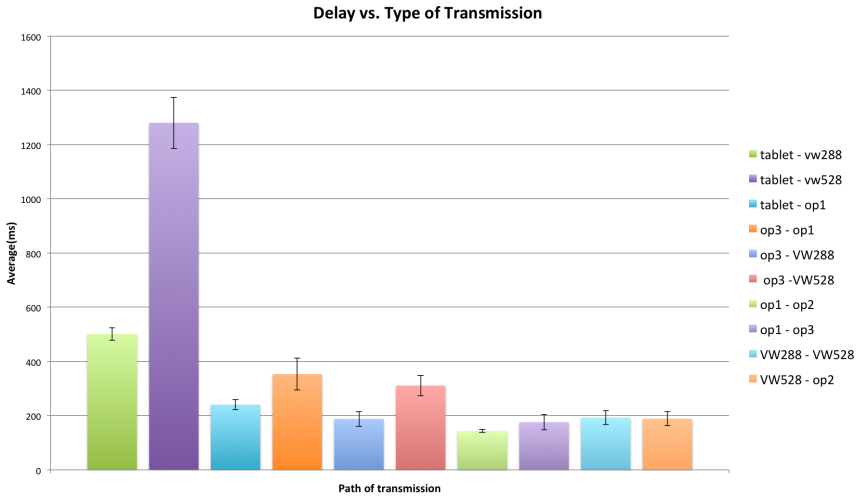


Figure 5.19: Delay versus Path of transmission

From this graph it is possible to see that most of the results are up to 200 millisec-

onds. The results which differs from this value can be explained with some interference (from the walls in laboratory as well as due to people passing between the boards during the tests) or congestion in the networks involved.

5.3 Harbor Tests

When everything worked properly in the laboratory, it was time to move to the testbed shown in Figure 5.20, testing the IBR-DTN implementation in a real network and with more vehicles.

In Oporto Harbor testbed the tests were carried out with 28 nodes, where 3 of them were RSUs and the others were OBUs, as it is represented in Figures 5.20 and 5.21. All nodes were intermittently connected with other vehicles and the infrastructure, and this was a perfect scenario to test the DTN mechanisms and, in particular, the implementation IBR-DTN in order to evaluate its performance with a larger number of nodes comparing to the tests performed in the laboratory.

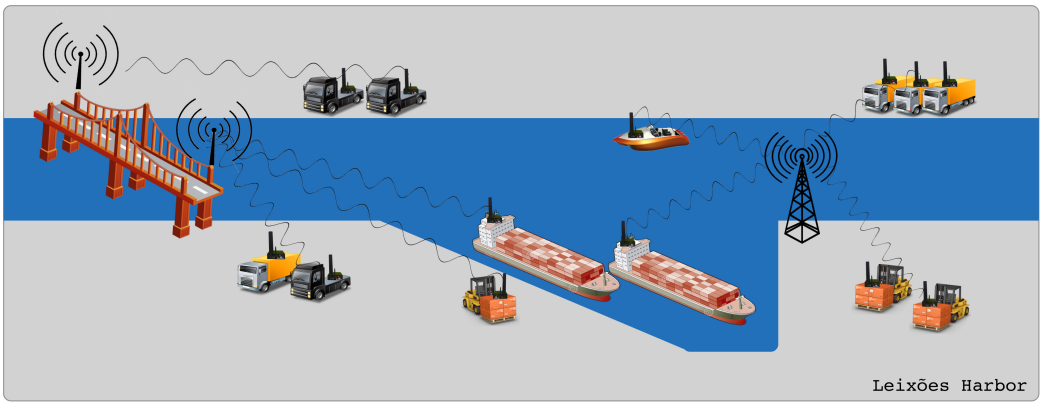


Figure 5.20: Harbor Scenario

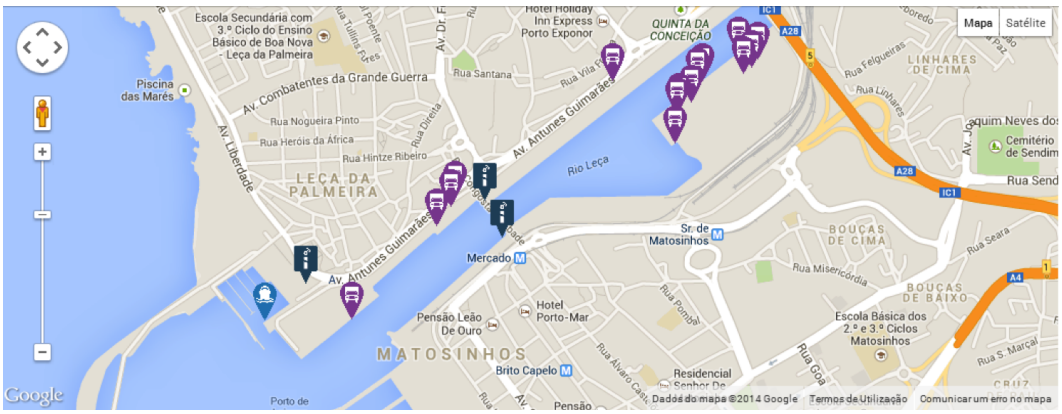


Figure 5.21: Harbor Map

The tests consisted on sending files while logging a set of information: the ID of the current board, the GPS coordinates, the size of bundles' folder and the date registered

each second. These files were named with the source board number plus the sequential number of the file. The files with this information were sent each 10 minutes from all OBUs installed in the trucks to the server placed in the Internet. In order to establish the contact between the vehicles and the server, other vehicles and infrastructure were used as relays that made use of the *store, carry and forward* DTN mechanism. There were other kind of files transmitted from all OBUs which contained the information about all the bundle's transmissions between nodes. Both files were transmitted via IEEE 802.11p in the network.

The nodes that store the bundles can save them for 12 hours, in case of the files with the measurements and for 48 hours in case of the logs with information of the transfers in the network. The files transmitted through DTN (either measurements or logs about transfers) have an average size of 35KB.

Three routing protocols are used for these tests: Prophet, Epidemic and Static routing.

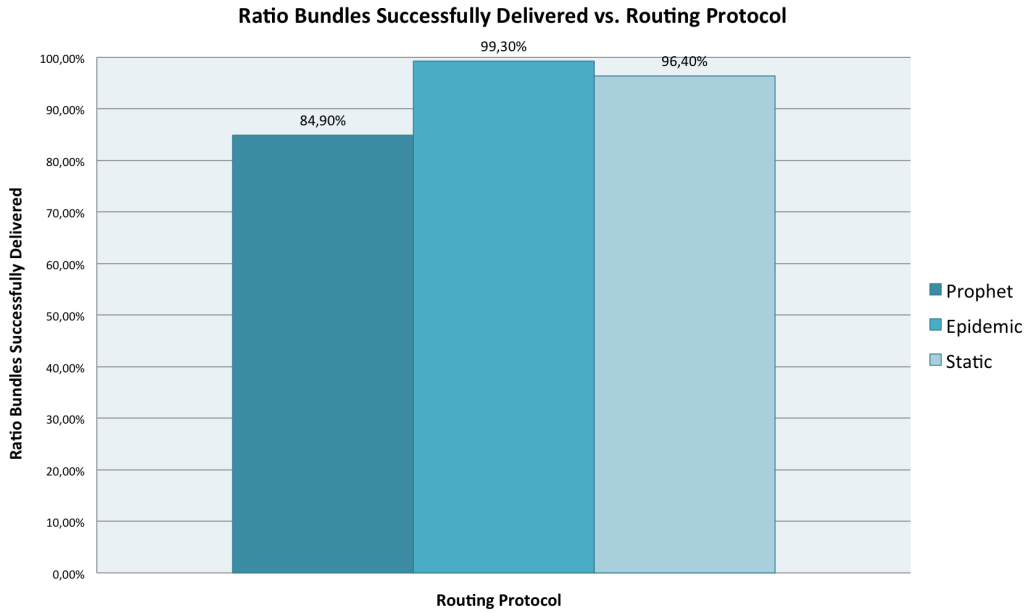


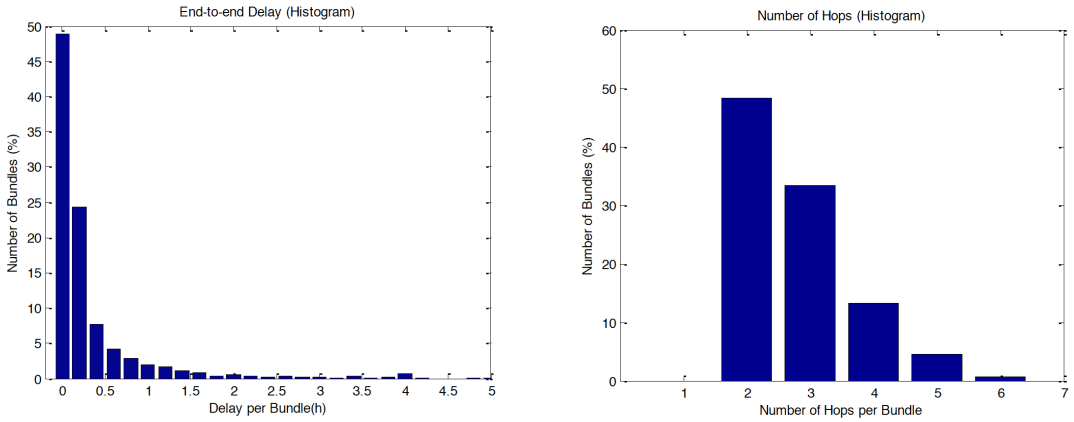
Figure 5.22: Ratio Bundles Successfully Delivered

Figure 5.22 shows the ratio of bundles successfully delivered for each one of the routing protocols used. As we can observe Prophet is the one with the lowest value. This can be explained by the mechanism used by this protocol which just sends bundles for "good" neighbors. This way, if these neighbors do not get in contact with the destination, the bundles are not delivered and this happened with about 15 percent of the bundles sent.

The Epidemic is the one with the best delivery ratio. This is explained by the transfer of the bundles to all neighbors the sender node gets in contact with. Not much lower than Epidemic, it is the Static protocol that also demonstrated a good performance, since this one does not overload the network as it happens with Epidemic. For this network, the Static protocol proves to be the most efficient and does not compromise the network in terms of storage and processing capacity.

In order to have a better analysis of the DTNs in a real platform of vehicles, the next graphs show the results for each metric studied.

The first test was performed with Prophet routing protocol and Figure 5.23 presents its results. These results include the transmission of files (with board ID, GPS coordinates, size of bundle's folder and date), as well as the logs to get the information about the sending of these files.



(a) Delay vs. Routing Protocol

(b) Number of Hops vs. Routing Protocol

Figure 5.23: Prophet Results

In Figure 5.23(a) it is presented the End-to-End (E2E) delay, i.e., the time it takes to send the files from a certain node until the server. It is possible to observe that most of the bundles take less than half an hour to reach the server, and just a minor part take more than one hour.

As the vehicles just send messages to the neighbor that has a large probability of meeting the final destination, sometimes the bundles are kept in the nodes too much time until that node appears. This fact causes a global higher delay and, although in a minor part, some bundles have taken more than 1 hour to be delivered. This can be explained because some of the trucks are frequently between cargo containers that do not allow the contact with infrastructure immediately.

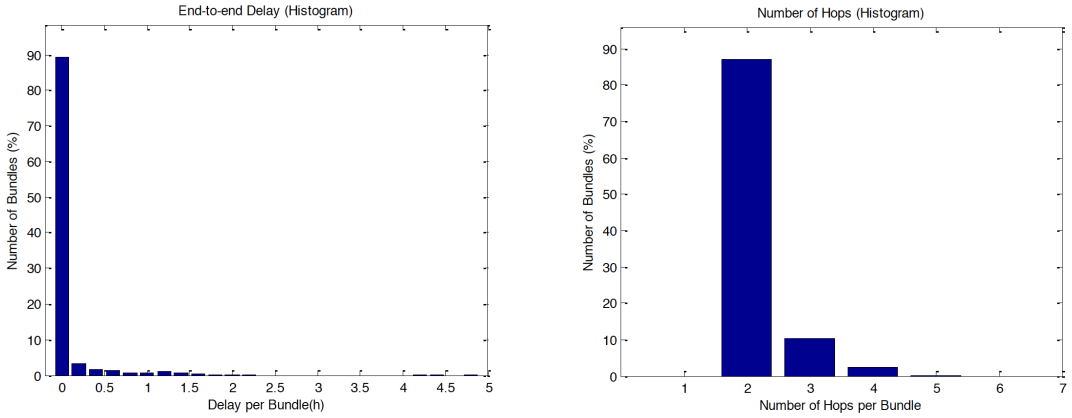
Figure 5.23(b) presents the number of hops, i.e., the number of nodes that receive the messages until they reach the final destination (the sender is not included). In the histogram the number 1 is null because RSUs themselves did not send any files to the server, they just forwarded them. The histogram starts in number 2 because the messages always needed at least 2 hops to reach the server: from sender vehicle to RSU and from RSU to server.

It is possible to observe that most of the bundles are delivered passing through the smallest number of hops. It is also possible to conclude that only less than 10 percent of bundles were sent through more than 4 hops.

These and other metrics are compared below considering the several protocols used.

The second protocol to be used was the Epidemic routing protocol whose results are presented in the next Figure 5.24. They also include the transmission of files (with

board ID, GPS coordinates, size of bundle's folder and date) as well as the logs to get the information about the transmission of these files.



(a) Delay vs. Routing Protocol

(b) Number of Hops vs. Routing Protocol

Figure 5.24: Epidemic results

In Figure 5.24(a) it is presented the E2E delay. We can observe that the time it takes to send the files from a certain node until the server decreased significantly, comparing it to the PROPHET protocol. Most of the bundles take less than 10 minutes to reach the final destination, and just a minor part takes more time.

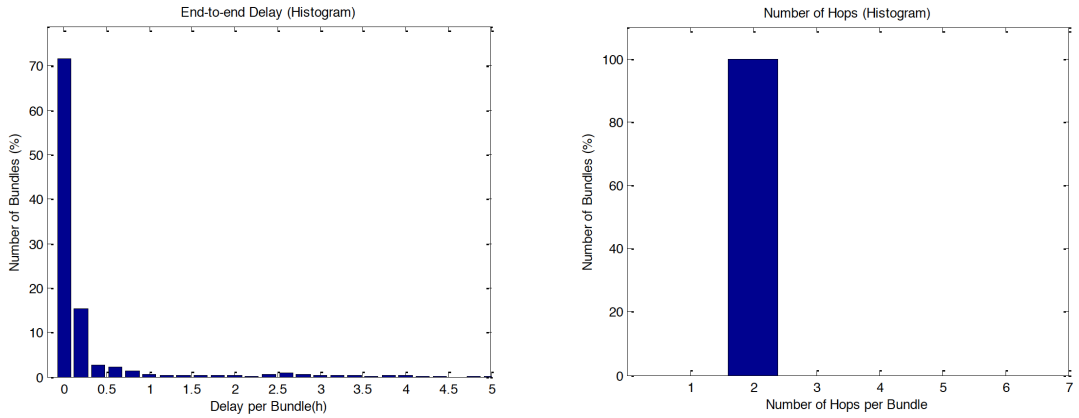
Figure 5.24(b) presents the number of hops from the source until the final destination: the most common number of hops is 2, the minimum possible because the bundles need to go always through the RSU. Nearly 90 percent of the bundles were delivered to the final destination just with 2 hops. There are also cases in which there are more hops, but those represent just a small part in all transfers.

Comparing to the Prophet routing protocol, there is no doubt that the global number of hops on the transferring process decreased, i.e., the path of bundles to reach the final destination involved less nodes.

The third protocol to be used was the Static routing protocol whose results are presented in the next Figure 5.25. In this protocol the files were sent from each OBU with predefined routes to the RSUs; this way, the next node is always a RSU.

Figure 5.25(a) shows the E2E delay: it is possible to observe that the bundles in their large majority were sent in less than 10 minutes. The other values can be explained by having into account that some vehicles are out of the range of the RSU for a long time (due to cargo containers and other obstacles). Once the vehicles just forward the bundles to the RSUs, sometimes they need to keep the bundles stored in their memory until a RSU becomes available in its range, and this can take some time, delaying the delivery to the final destination.

Figure 5.25(b) represents the number of hops the bundle traverse until the final destination. Since the OBU just sent its bundles to the RSU and the RSU is always connected to the Internet, and consequently to the server (final destination), it was expected the number of hops to be always 2.



(a) Delay vs. Routing Protocol

(b) Number of Hops vs. Routing Protocol

Figure 5.25: Static Results

The next graphs compare the different routing protocols (Prophet, Epidemic and Static) in 5 different metrics:

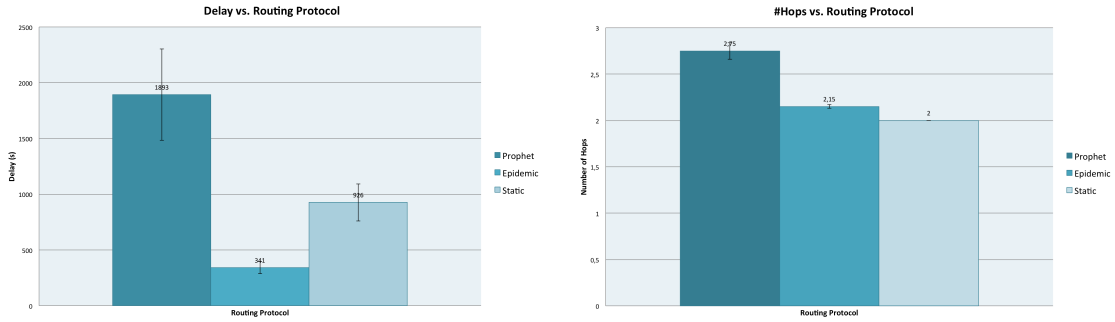
- End-to-End delay per bundle.
- Number of Hops per bundle.
- Total transmissions per bundle.
- Total replicas per bundle.
- Useless Replicas per bundle.

Figure 5.26(a) represents the E2E delay: it is possible to conclude that within the routing protocols tested, the Prophet was the one that presented lower performance, i.e., the time the bundle takes from the source node until the final destination is higher in Prophet than in Epidemic and Static. In Prophet the node just forwards the bundle to the node that has a higher probability of finding the destination or to another neighbor that also has a higher probability. The highest column shown in the graph can be explained by the lack of nodes with this high probability whenever a node has bundles to forward. This implies to store the bundle until a "good" neighbor is available which takes some time and so increases the delay.

The second routing protocol that takes more time is the Static one. This can be explained by the lack of RSUs in the moment one node has a bundle to forward: the node has to store the bundle until it gets into the range of an RSU.

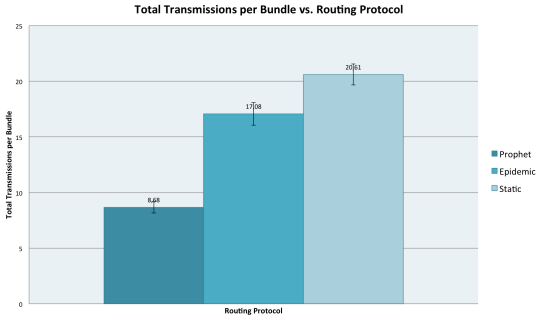
The Epidemic is the one that showed better performance, since it takes the lowest time sending bundles from the OBUs to the server. The lowest column demonstrates the epidemic mechanism, i.e., *all* nodes are contaminated with the bundle, which increases the probability of reaching the final destination.

Figure 5.26(b) shows the number of hops the bundles traverse until the final destination. In this metric, as expected, the Static routing always uses 2 hops per bundle.

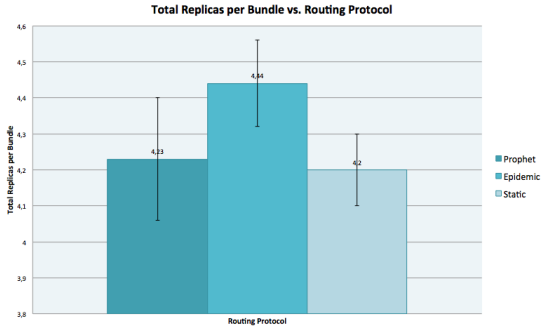


(a) Delay vs. Routing Protocol

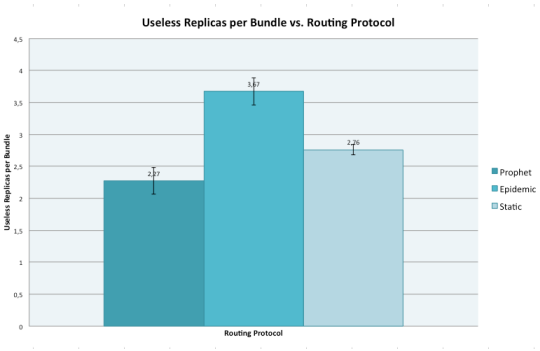
(b) Number of Hops vs. Routing Protocol



(c) Total Transmissions vs. Routing Protocol



(d) Total Replicas vs. Routing Protocol



(e) Useless Replicas vs. Routing Protocol

Figure 5.26: Analysis of different metrics for different routing protocols (Prophet, Epidemic ans Static)

Next, the Epidemic has an average of 2.15 hops per bundle. The protocol with the highest number of hops is the Prophet because it just sends the bundles to the nodes with higher probability in order to find the destination. This implies that the bundle may be forwarded through more nodes.

Figure 5.26(c) shows the total transmissions per bundle. This metric is important to show how a routing protocol spreads the bundles, and the way how they infect the network. Prophet showed the best result for this metric by having the lowest value. It can be explained by the way Prophet works. In this protocol the bundles are forwarded to other nodes that have a larger probability to find the destination than the node itself

does. This way there will not be many transmissions until finding the destination. The same does not happen with the Epidemic protocol: in this one, the number of nodes infected with the bundle is supposed to be much larger because of its epidemic mechanism. In Static routing the total transmissions per bundle were supposed to be 2 because the bundle just passes through three nodes. The high value observed is the result of a lack of acknowledgement of the received bundle by the receiver node. This way the bundle is constantly retransmitted by the source node increasing this metric.

Figure 5.26(d) shows the total replicas per bundle for the different routing protocols tested, i.e., the number of copies of the bundles. It was expected that the Epidemic protocol was the one that had the highest value because it forwarded the message to more nodes than the other routing protocols. The Prophet, forwards the bundles just for "good" neighbors and has a lower value. Concerning the Static routing, it follows the same explanation as in the 2 previous graphs.

Figure 5.26(e) shows the useless replicas per bundle, i.e., the copies of the bundles that were not coincident with the path of the bundle from the source to the destination. In the Prophet the replicas were, as expected, too low because the bundles were forwarded just to efficient nodes, i.e., to nodes that had higher probability of delivering the node to the destination. This way, almost all replicas of the bundles were useful. In the Epidemic protocol, as explained above, the epidemic mechanism made the bundle spread through too many nodes. Some of them were not efficient to deliver the bundle to the destination which causes the uselessness of the bundles. That is why this value is too high as the graph shows. The Static routing follows the same explanation as in the 3 previous graphs.

5.4 Summary

After solving the problems and overpassed difficulties mentioned in chapter 4, IBR-DTN was tested in both laboratory and in a testbed in Leixões harbor.

Several tests were performed in laboratory to evaluate the different scenarios a node can face: direct transfer with and without delay, indirect transfer (multi-hop) and indirect transfer with delay, that corresponds to the storage and carrying of bundles.

The results obtained in laboratory allowed a better understanding of IBR-DTN implementation: how it works and how it behaves in the different scenarios. The results showed a high performance of IBR-DTN for the different sizes of files sent, and the bundles were always successfully delivered. Nevertheless, when the bundle is forwarded in the relay node, it was detected a *pre-send phase* that was expected to happen just in the sender node. This phase involves several actions that make the process more slowly and consequently cause a higher delay.

In the section 5.2.4, we presented the results generated for the integration in a heterogeneous network, which allowed to conclude that IBR-DTN is a multi-platform implementation, i.e., it is available for the different OSs tested.

After the laboratory work, it was time to move to the testbed in Leixões harbor in order to test IBR-DTN with a higher number of nodes. Different tests with different routing protocols (Prophet, Epidemic and Static) were performed in order to decide which of them was the most suitable and efficient. Depending on the amount of storage in the boards, the quantity of nodes to reach in the network and how urgent the

information to be transmitted is, we can choose the most appropriated routing protocol. Nevertheless, it is possible to affirm that the tested routing protocols do not have into account some essential characteristics of VANETs, like direction of movement, time-changing connectivity, density of vehicles, speed or probability to find RSUs. Therefore, there is the need to create a protocol adapted to the characteristics of real-world mobile networks.

Chapter 6

Conclusion and Future Work

6.1 Conclusions

The main aim of this dissertation was to study the DTN mechanisms, to implement it in a vehicular environment and test it in a heterogeneous network, proving that it works in a multi-platform system with different Operative Systems.

In this dissertation, several DTN implementations and routing protocols were studied, and IBR-DTN was extended to work in heterogeneous environments with different OS and network technologies, and also in vehicular environments. This extension required the solution of several problems detected in the implementation during the integration in the boards both in the laboratory and in the testbed.

This dissertation focuses on the real-world tests and the discussion of the results. After having installed and tested IBR-DTN in all boards, some tests were carried out in laboratory having the boards without movement. These tests were made to evaluate IBR-DTN performance in the different scenarios a node can meet: direct transfer with and without delay, indirect transfer (multi-hop) and indirect transfer with delay, which corresponds to the storage and carrying of bundles. The tests were all carried out for different sizes of files and by changing the number of nodes in the network.

We have shown that IBR-DTN works well in the vehicular OBUs and RSUs. Nevertheless, it was detected a problem that may delay the global process of a transfer when this is not direct, i.e., it was detected that when we have a relay in the transfer, it also happens a *pre-send phase* (where it is performed once again the bundle' creation, neighbors discovery, storage in the bundle's folder, etc). This phase was expected to run only in the sender node. In this case it can be considered as a waste of time in the transferring process.

Still in the laboratory, it was tested IBR-DTN in a heterogeneous network with servers, OBUs, Single Board Computers (SBCs), tablet, Raspberry Pi and Macbook which had different OSs (Openwrt, Ubuntu, Debian, MacOS and Android). The tests consisted on sending files from one device to another. First it was tested the sending of files from the boards to the servers and then just between boards. All these transfers were aimed to cross several networks from different types (using IEEE 802.11p and IEEE 802.11g and even Ethernet). All results showed a good behavior, and all files sent were received in the predefined destination. This way, it is possible to conclude that IBR-DTN was successfully integrated in a heterogeneous network, having a good

operation in the tests.

After the laboratory, it was time to move to a real-world testbed. In Leixões Harbor the tests consisted on sending files with the ID of the current board, the GPS coordinates, the size of bundles' folder and the current date. This information was registered in a file each second, and it was sent each 10 minutes from all the OBUs installed in the trucks to the server placed in the Internet. The average size of these files was 35KB. In order to establish the contact between the vehicles and the server, other vehicles and infrastructure were used as relays which made use of the *store, carry and forward* DTN mechanism.

Three routing protocols were tested in this platform: Prophet, Epidemic and Static. It was shown that the Prophet routing protocol uses too much CPU on the process of exchanging, between vehicles, the values of probability of nodes to find the destination. This protocol also demonstrated to be the one that took more time in E2E transfers. In spite of that, it was the one that had the lowest number of transmissions, which is an advantage in the storage capacity, i.e., as the boards have low storage capacity: if the number of transmissions is small, the number of bundles stored in the boards will be small too. Unexpectedly, PRoPHET shows much higher delay than static routing, which means that nodes are not transmitting whenever they see an RSU. This happens due to the PRoPHET's method for calculation of contacts between nodes, which is not designed to deal with nodes with permanent connectivity. Thus, we see DTN routing protocols should work not only in intermittently connected network, but also in permanent connected networks and their interfaces.

On the other hand, the Epidemic routing protocol, although with lower E2E delay, floods the network and exceeds the storage capacity of the boards, as expected. As it was possible to observe, the number of useless replicas is very high. This way, we can conclude that this protocol does not have a high efficiency.

The Static routing was the one that revealed the best performance, taking into account the ratio of bundles successfully delivered and the load of the network. This protocol only sent files from OBUs when they were in direct contact with RSUs. This means that there were no copies spread in the network. This fact makes this protocol the most efficient comparing it to the others tested. However, this just hapenned because the network is well covered with RSUs and the trucks get in contact with RSUs frequently: this is not the case in other scenarios, and a DTN protocol in vehicular environments will require the use of other vehicles as mules for the delivery of information. Moreover, it needs to be improved by adding acknowledgments whenever bundles reach the destination. This will avoid the sender to send several copies. We also observed that transmissions kept occurring even after bundles had already been received by relaying nodes or by the destination, which lead to wasting many resources. An efficient DTN routing protocol could eventually prevent this using an acknowledgement system for bundle delivery. Such acknowledgement strategy should be designed for situations where acknowledgments are viable and useful, and preferably in a manner that is optimized for the routing protocol.

To summarize, we can conclude that we got very good results in the ratio of bundles successfully delivered, which proves that DTN is a good competitor to the use of cellular network on transmitting non urgent information. Nevertheless, we can also conclude that DTN routing strategies should take into account the nodes' pattern of interaction

and movements. In a case like this, with a lack of patterns and focus on delivering data through fixed RSUs, a geographical approach would be more adequate.

Following the results of this work, we believe that there is the opportunity for the proposal and evaluation of DTN routing algorithms that include an acknowledgement dissemination strategy, and that take into account characteristics of real-world mobile networks which are often ignored: time-changing connectivity, density of vehicles, probability to find road side units, speed, direction, specific patterns of mobility, disrupted-connected interfacing, in-node processing delays, among others.

6.2 Future Work

Since DTN is still an object of study in several research groups, a lot of work can be done in order to improve vehicular networks. Concerning the work developed for this dissertation and everything we have mentioned previously, we would like to suggest several points we think should be developed as future work:

- To analyze the code of IBR-DTN in order to know why there is a *pre-send phase* in relay nodes. Removing this phase, the global E2E delay of the transfer would be much lower.
- To add Acknowledgment dissemination strategy to the Static routing protocol in order to inform the sender node that the bundles reached the destination. This will avoid the current problem of this protocol when it is always sending the bundles, even when they have already reached the final destination.
- To build a new routing protocol based on vehicular metrics having into account the density of vehicles in a given area, speed, direction, etc.
- To implement more RSUs in areas where they do not exist, so that the traffic of bundles may flow until the servers placed in the Internet which are accessible via infrastructures (RSUs). This would also reduce the expiration of bundles.
- The tests in the harbor proved the IBR-DTN functionality in vehicular environment, but tests need to be done using a larger network with a considerable larger number of nodes evaluating its behavior. It would be very useful if they were tested in the larger environment of a city.

All these improvements would bring more confidence, reliability and applicability in VANETs, and they would be a big step to improve people's quality of life mainly in what their driving experience is concerned.

Bibliography

[1] "Overall Growth of Data Traffic," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf, accessed in May, 2014.

[2] "The Future of Intelligent Transport Systems," http://mubbisherahmed.wordpress.com/2011/11/29/the-future-of-intelligent-transport-systems-its/, accessed in May, 2014.

[3] "Vehicular Network communication types," http://adrianlatorre.com/projects/pfc/img/vanet_full.jpg, accessed in May, 2014.

[4] D. Jiang and L. Delgrossi, "Ieee 802.11p: Towards an international standard for wireless access in vehicular environments," in *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, May 2008, pp. 2036–2040.

[5] R. Alves and I. Campbell, "Redes veículares: Princípios, aplicações e desafio (book), chapter 5," May, 2009, in Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2009.

[6] H. Moustafa and Y. Zhang, *Vehicular Networks: Techniques, Standards and Applications*, Chapter Routing in Vehicular Networks: A User's Perspective, 2009, cRC Press.

[7] E. P. Jones and P. A. Ward, "Routing strategies for delay-tolerant networks," *ACM Computer Communication Review (CCR)*, 2006.

[8] K. Fall and S. Farrell, "Dtn: an architectural retrospective," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 828–836, 2008.

[9] K. William, "The Future Is Ubiquitous - Delay Tolerant Networks - University of Missouri-rolla," http://web.mst.edu/~mobildat/DTN/index.html, May 2006.

[10] A. Haris, "A dtn study: Analysis of implementations and tools," Ph.D. dissertation, Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2010.

[11] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *Service Assurance with Partial and Intermittent Resources*. Springer, 2004, pp. 239–254.

[12] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[13] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A survey of inter-vehicle communication protocols and their applications," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 2, pp. 3–20, 2009.

[14] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. C. Wolf, "Ibr-dtn: A lightweight, modular and highly portable bundle protocol implementation," *Electronic Communications of the EASST,*, pp. 1–11, January 2011.

[15] "Cambria GW2358-4 Single Board Computer," http://www.gateworks.com/product/item/cambria-gw2358-4-network-processor, accessed in June, 2014.

[16] "Raspberry Pi," http://www.raspberrypi.org, accessed in June, 2014.

[17] "OpenWRT Development Guide," http://www.ccs.neu.edu/home/noubir/Courses/CS6710/S12/material/OpenWrt_Dev_Tutorial.pdf, February 2012.

[18] P. R. Pereira, A. Casaca, J. J. P. C. Rodrigues, V. N. G. J. Soares, J. Triay, and C. Cervello-Pastor, "From delay-tolerant networks to vehicular delay-tolerant networks," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, pp. 1166–1182, Fourth 2012.

[19] D. Evans, "The Internet of Things - How the Next Evolution of the Internet Is Changing Everything (Cisco)," http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, pp. 2–4, April, 2011, accessed in July, 2014.

[20] M. Nekovee, "Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and grids," in *Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and grids, Edinburgh, UK*, 2005.

[21] H. Moustafa and Y. Zhang, *Vehicular Networks: Techniques, Standards and Applications*, Chapter Applications and Services, 2009, cRC Press.

[22] "General Motors Corporation," http://www.onstar.com, accessed in December 1, 2008.

[23] "Ieee standard for wireless access in vehicular environments (wave) - networking services," *IEEE Std 1609.3-2010 (Revision of IEEE Std 1609.3-2007)*, pp. 1–144, Dec 2010.

[24] "Ieee standard for wireless access in vehicular environments (wave)–multi-channel operation," *IEEE Std 1609.4-2010 (Revision of IEEE Std 1609.4-2006)*, pp. 1–89, Feb 2011.

[25] M. Javed and J. Khan, "A geocasting technique in an ieee802.11p based vehicular ad hoc network for road traffic management," in *Australasian Telecommunication Networks and Applications Conference (ATNAC), 2011*, Nov 2011, pp. 1–6.

[26] "802.11p / DSRC / WAVE," http://www2.litepoint.com/solutions/802-11p, accessed in July, 2014.

[27] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments," *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, pp. 1–51, July 2010.

[28] H. Moustafa and Y. Zhang, *Vehicular Networks: Techniques, Standards and Applications*, Chapter Introdution to Vehicular Networks, 2009, cRC Press.

[29] A. Chatterjee, H.-W. Kaas, T. Kumaresh, and P. J. Wojcik, "A road map for telematics: Automakers betting on telematics face a highly uncertain market. they should focus on building great cars while choosing their telematics investments carefully," *The McKinsey Quarterly*, p. 100, 2002.

[30] "Car 2 Car Communication Consortium," http://car-2-car.or, accessed in December 2, 2008.

[31] B. Paul, M. Ibrahim, M. Bikas, and A. Naser, "Vanet routing protocols: Pros and cons," *arXiv preprint arXiv:1204.1201*, 2012.

[32] C. Mbarushimana and A. Shahrabi, "Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, vol. 2. IEEE, 2007, pp. 679–684.

[33] R. Jain, A. Puri, and R. Sengupta, "Geographical routing using partial information for wireless ad hoc networks," *Personal Communications, IEEE*, vol. 8, no. 1, pp. 48–57, 2001.

[34] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad hoc networks*, vol. 3, no. 3, pp. 325–349, 2005.

[35] B. Tavares, "Opportunistic transmission of information in vehicular networks," December, 2013, dissertação para obtençao do grau de Mestrado em Engenharia Electrónica e Telecomunicações na Universidade de Aveiro em parceria com o Instituto de Telecomunicações.

[36] M. Menouar, M. Lenardi, and F. Filali, "A movement prediction based routing protocol for vehicle-to-vehicle communications," *Communications*, vol. 21, pp. 07–2005, 2005.

[37] V. Naumov and T. Gross, "Connectivity-aware routing (car) in vehicular ad-hoc networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, May 2007, pp. 1919–1927.

[38] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, "Geographic routing in city scenarios," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 1, pp. 69–72, 2005.

[39] I. Leontiadis and C. Mascolo, "Geopps: Geographical opportunistic routing for vehicular networks," in *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, June 2007, pp. 1–6.

[40] J. LeBrun, C.-N. Chuah, D. Ghosal, and M. Zhang, "Knowledge-based opportunistic forwarding in vehicular wireless ad hoc networks," in *Vehicular technology conference, 2005. VTC 2005-Spring. 2005 IEEE 61st*, vol. 4. IEEE, 2005, pp. 2289–2293.

[41] B. chong Seet, G. Liu, B. sung Lee, C. heng Foh, and K. kee Lee, "A-star: A mobile ad hoc routing strategy for metropolis vehicular communications," in *Proc. NETWORKING 2004*, 2004, pp. 989–999.

[42] J. Zhao and G. Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," *Vehicular Technology, IEEE Transactions on*, vol. 57, no. 3, pp. 1910–1922, 2008.

[43] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 27–34.

[44] "InterPlanetary Internet (IPN) Internet Project," http://www.ipnsig.org, accessed in July, 2014.

[45] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, "Delay-tolerant networking: an approach to interplanetary internet," *Communications Magazine, IEEE*, vol. 41, no. 6, pp. 128–136, 2003.

[46] C. Caini, P. Cornice, R. Firrincieli, and D. Lacamera, "A dtn approach to satellite communications," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 820–827, 2008.

[47] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-tolerant networking architecture," *RFC4838, April*, 2007.

[48] K. L. Scott and S. Burleigh, "Rfc 5050 - bundle protocol specification," November, 2007, nASA Jet Propulsion Laboratory.

[49] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless communications, IEEE*, vol. 11, no. 6, pp. 6–28, 2004.

[50] J. Burgess, B. Gallagher, D. Jensen, and B. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–11.

[51] A. Balasubramanian, B. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 373–384, Aug. 2007. [Online]. Available: http://doi.acm.org/10.1145/1282427.1282422

[52] H. S. Bindra and A. Sangal, "Performance comparison of rapid, epidemic and prophet routing protocols for delay tolerant networks," *International Journal of Computer Theory and Engineering*, vol. 4, no. 2, 2012.

[53] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, ser. WDTN '05. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[54] M. Zhang and R. Wolff, "Routing protocols for vehicular ad hoc networks in rural areas," *Communications Magazine, IEEE*, vol. 46, no. 11, pp. 126–131, November 2008.

[55] ——, "A border node based routing protocol for partially connected vehicular ad hoc networks," *Journal of Communications, Academy Publisher*, vol. 5, no. 2, pp. 130–143, February 2010.

[56] N. Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz, "Routing in sparse vehicular ad hoc wireless networks," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 8, pp. 1538–1556, Oct 2007.

[57] R. Bishop, "A survey of intelligent vehicle applications worldwide," in *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*. IEEE, 2000, pp. 25–30.

[58] J. Chennikara-Varghese, W. Chen, O. Altintas, and S. Cai, "Survey of routing protocols for inter-vehicle communications," in *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on.* IEEE, 2006, pp. 1–5.

[59] S. Tsugawa, "Inter-vehicle communications and their applications to intelligent vehicles: an overview," in *Intelligent Vehicle Symposium, 2002. IEEE*, vol. 2. IEEE, 2002, pp. 564–569.

[60] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, and S. Keshav, "Very low-cost internet access using kiosknet," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 5, pp. 95–100, Oct. 2007. [Online]. Available: http://doi.acm.org/10.1145/1290168.1290181

[61] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden, "Cartel: A distributed mobile sensor computing system," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 125–138. [Online]. Available: http://doi.acm.org/10.1145/1182807.1182821

[62] S. Lahde, M. Doering, W.-B. Pöttner, G. Lammert, and L. Wolf, "A practical analysis of communication characteristics for mobile and distributed pollution measurements on the road: Research articles," *Wirel. Commun. Mob. Comput.*, vol. 7, no. 10, pp. 1209–1218, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1002/wcm.v7:10

[63] S. Lahde, M. Doering, W.-B. Pöttner, and G. Lammert, "Mobile and distributed measurement of air pollution in metropolitain areas using car2x techniques."

[64] "How TomTom's HD Traffic and IQ Routes data provides the very best routing," http://www.tomtom.com/lib/doc/download/HDT_White_Paper.pdf, 2009.

[65] H. Soroush, N. Banerjee, A. Balasubramanian, M. D. Corner, B. N. Levine, and B. Lynn, "Dome: A diverse outdoor mobile testbed," in *Proceedings of the 1st ACM International Workshop on Hot Topics of Planet-Scale Mobility Measurements*, ser. HotPlanet '09. New York, NY, USA: ACM, 2009, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/1651428.1651431

[66] A. Balasubramanian, B. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 373–384, 2007.

[67] V. Soares, F. Farahmand, and J. Rodrigues, "A layered architecture for vehicular delay-tolerant networks," in *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, July 2009, pp. 122–127.

[68] K. Scott, "Disruption tolerant networking proxies for on-the-move tactical networks," in *Military Communications Conference, 2005. MILCOM 2005. IEEE*, Oct 2005, pp. 3226–3231 Vol. 5.

[69] "Car 2 car Communication consortium manifesto," http://elib.dlr.de/48380/1/C2C-CC_manifesto_v1.1.pdf, 2007.

[70] "Licklider Transmission Protocol," http://tools.ietf.org/html/rfc5325, accessed in September, 2008.

[71] “Interplanetary Overlay Network,” http://sourceforge.net/projects/ion-dtn/, accessed in July, 2014.

[72] “Bytewalla DTN on Android Phones,” http://sourceforge.net/projects/bytewalla, accessed in July, 2014.

[73] “DT-Talkie Architecture,” http://www.netlab.tkk.fi/tutkimus/dtn/dttalkie, accessed in 2014.

[74] M. Blanchet and S. Perreault, “Postellation: A DTN Implementation,” https://www.ietf.org/proceedings/79/slides/DTNRG-3.pdf, 2010.

[75] “DTN2,” http://www.dtnrg.org/docs/code/DTN2/doc/manual/intro.html, accessed in 2014.

[76] “The Delay-Tolerant Networking Research Group ,” http://www.dtnrg.org/wiki, accessed in May 2014.

[77] “OpenWRT - Wireless Freedom,” https://openwrt.org, accessed in May 2014.

[78] “A C library for embedded Linux,” http://www.uclibc.org, accessed in May 2014.

[79] “DTN IP Neighbor Discovery (IPND),” http://tools.ietf.org/html/draft-irtf-dtnrg-ipnd-02, accessed in November, 2012.

[80] “libcurl - the multiprotocol file transfer library,” http://curl.haxx.se/libcurl/, accessed in March, 2014.

[81] A. Vahdat, D. Becker *et al.*, “Epidemic routing for partially connected ad hoc networks,” Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[82] “SQLite Database - Software Library,” http://www.sqlite.org, accessed in July, 2014.

[83] “Raspberry Pi Foundation - documentation,” http://www.raspberrypi.org/documentation/, accessed in May, 2014.

[84] “OpenWRT Buildroot,” http://wiki.openwrt.org/about/toolchain, accessed in May 2014.

[85] “OpenWRT - Creating Packages,” http://wiki.openwrt.org/doc/devel/packages, accessed in April 2014.

[86] “IBR-DTN Source code,” http://trac.ibr.cs.tu-bs.de/project-cm-2012-ibrdtn/wiki/source, accessed in February 2014.

[87] “IBR-DTN ChangeLog,” https://trac.ibr.cs.tu-bs.de/project-cm-2012-ibrdtn/wiki/changelog, accessed in April 2014.

[88] S. Schildt, T. Lorentzen, J. Morgenroth, W.-B. Pöttner, and L. Wolf, “Free-riding the bittorrent dht to improve dtn connectivity,” in *Proceedings of the Seventh ACM International Workshop on Challenged Networks*, ser. CHANTS ’12. New York, NY, USA: ACM, 2012, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/2348616.2348619

[89] “PTPD - Precision Time Protocol Daemon,” http://sourceforge.net/p/ptpd/wiki/Home/, November, 2013, accessed in July, 2014.

[90] "NTP - Network Time Protocol," hhttp://www.ntp.org, February, 2014, accessed in July, 2014.

[91] C.-C. Chao, S.-P. Huang, and H.-L. Hung, "Embedded system on ntp," in *Computer Sciences and Convergence Information Technology, 2009. ICCIT '09. Fourth International Conference on*, Nov 2009, pp. 852–857.

[92] S.-M. Jun, D.-H. Yu, Y.-H. Kim, and S.-Y. Seong, "A time synchronization method for ntp," in *Real-Time Computing Systems and Applications, 1999. RTCSA '99. Sixth International Conference on*, 1999, pp. 466–473.